IPG Politécnico da Guarda
Polytechnic of Guarda

RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

Gilberto Soeiro Cardoso

junho | 2016

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

RELATÓRIO DE PROJETO
GESTÃO DE ENERGIAS RENOVÁVEIS

Gilberto Soeiro Cardoso

RELATÓRIO PARA A OBTENÇÃO DO GRAU DE LICENCIADO

EM ENGENHARIA INFORMÁTICA

Junho/2016

Projeto de Informática

2015/2016

Gestão de Energias Renováveis

Orientador: Prof. Luís Figueiredo

Realizado por: Gilberto Soeiro Cardoso

Número de aluno: 1011314

RESUMO

Este trabalho consiste no processo de desenvolvimento de uma solução viável para um problema real dos sistemas de produção de energia renováveis, em que muitas das vezes a energia produzida não é devidamente aproveitada. Já existem algumas soluções no mercado, mas com conceitos de funcionamento diferentes.

A ideia do autor é criar um sistema para armazenar a energia elétrica produzida em excesso na forma de frio e para isso utiliza a arca frigorífica.

Ao longo do trabalho foi desenvolvido um módulo de recolha de dados para posterior análise da viabilidade da ideia e, seguidamente, dois módulos que comunicam entre si.

O módulo principal utiliza como microcontrolador um Arduíno nano, conectado a um transmissor de comunicação RF 433MHz. Este módulo tem ainda dois sensores de corrente, o primeiro analisa quanto o sistema solar está a produzir e o segundo analisa quanto a habitação está a consumir. O módulo está previamente configurado com os valores em que ele deve dar ordens para o segundo módulo.

O segundo módulo emula um Termóstato da arca frigorífica, tem como microcontrolador também um Arduíno nano, possui um recetor de comunicação RF 433MHz e um sensor de temperatura. O módulo vai funcionar como um termostato, mantendo a temperatura da arca a um nível máximo seguro de congelamento (-14ºC), mas caso exista energia em excesso suficiente, recebe a ordem do primeiro módulo para trabalhar até atingir a temperatura mínima de congelamento (-24ºC)

A arca frigorífica é um eletrodoméstico que não está sempre em trabalho. Após alguma análise de um exemplo real, concluiu-se que passa mais de metade do dia em repouso. Com este sistema, ela é obrigada a trabalhar durante mais horas a consumir a energia solar que não esteja a ser aproveitada e consequentemente repousar mais durante o período em que a energia não seja da fonte solar.

ABSTRACT

This project consists in the development of a viable solution to the real problem of renewable energy production systems, in which, most of the time, the energy produced is not properly used. There are already some solutions on the market, but with different operating concepts.

The idea of the author is to create a system to store the electricity produced in excess in the form of cold by using a freezer. During the work was developed a data collection module to analyse the viability of the idea, and then two other modules communicating with each other.

The main module consists of a Arduino Nano microcontroller, connected to a communication 433MHz RF transmitter. This module also has two current sensors, the first analyses the amount of energy the solar system is producing and the second examines how the housing is consuming. This first module is pre-configured to give orders to the second module.

The second module emulates a freezer's Thermostat and consists of a Arduino Nano micro controller with a communication receiver 433MHz RF and a temperature sensor. The module will work as a thermostat, keeping the freezer's temperature to a maximum safe freezing level (-14º), but in case there is excess of energy it will receive an order from the first module to work until it reaches the minimum freezing temperature (-24º)

The freezer is an appliance that is not always running. After the analysis of an actual sample, it was concluded that the freezer is off during more than half a day. With this system, it is required to work longer hours to consume the excess of solar energy and consequently forced to stop during the period in which energy is not from the solar source.

ÍNDICE

LISTA DE FIGURAS

1. INTRODUÇÃO

O atual relatório caracteriza o desenvolvimento do projeto de gestão de energias renováveis, realizado pelo aluno Gilberto Soeiro Cardoso, no âmbito da unidade curricular de Projeto de informática, do 3º ano da Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda.

1.1. Contextualização

Com o avançar da tecnologia, houve um grande aumento de dispositivos eletrónicos, aumentando assim o consumo energético em grande escala como mostra a Figura 1. Mesmo nas próprias habitações, podemos observar cada vez mais dispositivos e por sua vez o consumo energético destes acumula-se. Com este cenário, cada vez mais se tem em conta a eficiência energética das habitações.

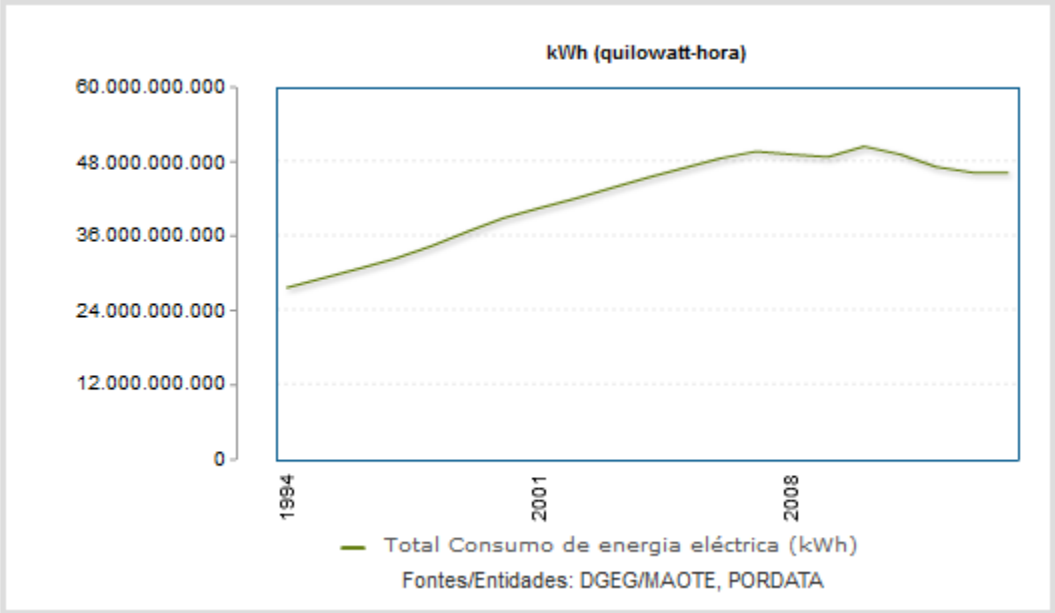

Figura 1 - Consumo Energia Elétrica em Portugal

A eficiência energética de uma habitação não é vista só na sua climatização (embora seja o maior peso) mas sim através de tudo o que sejam gastos de energia, principalmente energia elétrica. Estamos constantemente a ver avanços na tecnologia a fim de se conseguir melhores eficiências. Um grande exemplo disso é a iluminação onde antes era essencialmente halogéneo, foi seguida pelas lâmpadas "económicas" florescentes, e atualmente já se utiliza na maioria LED's, onde a sua eficiência é drasticamente superior ao halogéneo.

Mas como os consumos nas habitações não se ficam pela iluminação, hoje em dia nenhuma habitação vive sem uma televisão, frigorífico, micro-ondas e muitos outros dispositivos que consomem energia, levando a que mesmo com a evolução de eficiência em todos os dispositivos o consumo energético continua a ser elevado. Com este cenário existem várias ideias de tornar as habitações mais eficientes com o objetivo de as tornar o mais possível autossustentáveis. Já é usual ver painéis solares de águas quentes sanitárias nas habitações (AQS), passando já a ser obrigatório por lei a sua implementação nas construções atuais.

Atualmente, para além de se instalarem painéis de águas quentes sanitárias também se instalam painéis fotovoltaicos com o objetivo de produção de energia elétrica para a própria habitação. Existem algumas tipologias de instalação dos painéis fotovoltaicos. Presentemente, o que se instala é um conjunto de painéis dimensionado para os consumos da habitação, onde toda a energia é injetada diretamente na habitação. Este tipo de sistema tem a vantagem da sua instalação ser mais simples e barata, mas toda a energia que é produzida no momento e não é consumida é considerada perdida pois não há forma de a armazenar. (Nova Energia, s.d.)

Tendo como base esta tipologia surgiu a ideia de tentar melhorar um pouco o seu aproveitamento armazenando-a em forma de frio com a introdução de um módulo que monitoriza o consumo energético da habitação em si, sabendo assim se está a ser utilizada toda a energia produzida pelos painéis solares ou se está a haver perda de energia por saída para a rede. Caso haja energia que esteja a sair para a rede, existe a oportunidade de a aproveitar, para isso existe outro modulo desenhado especificamente para uma arca congeladora (eletrodoméstico que predomina em praticamente todas as habitações) que recebe ordens do primeiro módulo e assim consegue consumir alguma dessa energia. Pode-se considerar que a arca frigorífica vai funcionar como uma espécie de *bateria* para o excesso de produção. Este sistema cada vez mais é implementado, pois Portugal tem uma localização geográfica muito favorável, onde permite que

estes sistemas produzam energia entre 5 a 11 horas diárias em média ao longo do ano. (WeatherOnline, s.d.)

1.2. Objetivos do projeto

Este projeto pretende chegar a uma solução autónoma para melhorar a eficiência de um sistema de energias renováveis de autoconsumo numa habitação, não melhorando a sua produção do sistema em si, mas aperfeiçoando a maneira como essa energia está a ser consumida, compensando mais o utilizador. Para isso ser possível vai ser necessário analisar todos os sistemas envolventes e perceber como estão interligados.

A solução terá de ser autónoma, pois hoje em dia caminhamos para um mundo onde os dispositivos sabem tomar as suas próprias decisões sem ser preciso a intervenção de um utilizador. Tem também de ser uma solução simples para que os custos quer de produção, quer de aplicação sejam relativamente baixos.

Para ser mais versátil, irá tentar chegar a uma solução possível de ser aplicada na maioria das habitações atuais.

Tendo em conta os pontos referidos anteriormente, podemos estabelecer vários objetivos a cumprir:

1. Criar um dispositivo de recolha de dados de uma arca frigorífica para posterior analise
2. Analisar os dados para saber a viabilidade do projeto
3. Criar um módulo que monitoriza o consumo de energia
4. Criar um módulo de termóstato para arca frigorífica
5. Explorar a tecnologia do ESP8266

2. ESTADO DE ARTE

Este capítulo é dedicado a sistemas em que as suas funções se enquadram no esquema deste projeto, analisar as suas vantagens e desvantagens, podendo assim ter uma base sólida para a sua elaboração. Todas as informações relevantes foram pesquisadas pelo autor.

2.1. Inversores híbridos

Um inversor híbrido é um inversor onde todas as partes do sistema ficam conectadas a ele como é possível ver na Figura 2.

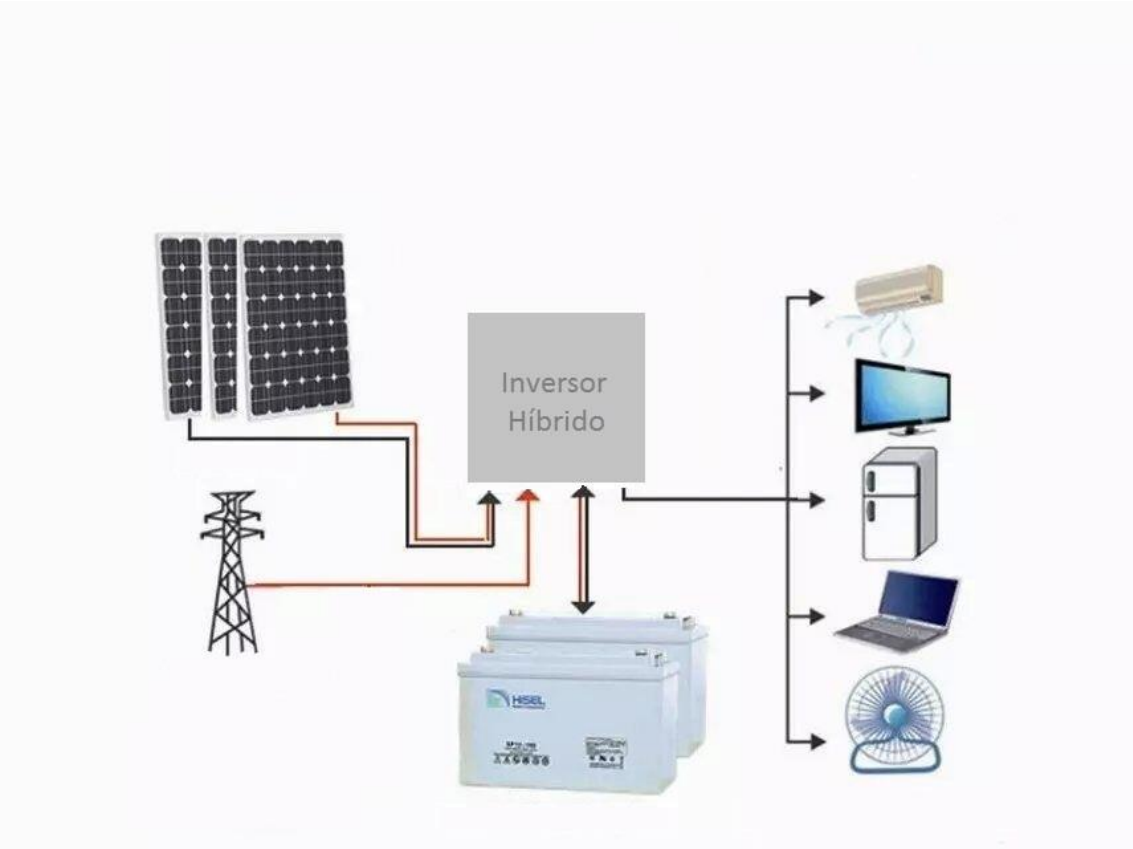

Figura 2 - Esquema de um Inversor Híbrido

Este tipo de inversor recebe energia das diversas fontes possíveis, seja ela da rede ou da fonte renovável (solar), mediante a sua configuração é possível definir no inversor para dar prioridade a energia renovável. O inversor alimenta assim a habitação com a energia solar, caso

esteja a produzir mais energia que a necessária para a habitação, essa energia é armazenada em baterias para poder ser consumida quando necessária, assim sendo o inversor só vai utilizar energia da rede quando as baterias estiverem esgotadas e não existir energia suficiente da fonte renovável. Este inversor elimina o problema de que a energia não consumida saia para a rede elétrica nacional, pois esta energia é canalizada para as baterias, mas existe a desvantagem de ficar um sistema muito mais dispendioso e com todas as desvantagens de utilização das baterias.

2.2. Inibidor de injeção na rede

Uma outra solução é um inibidor de injeção na rede como o apresentado na Figura 3.

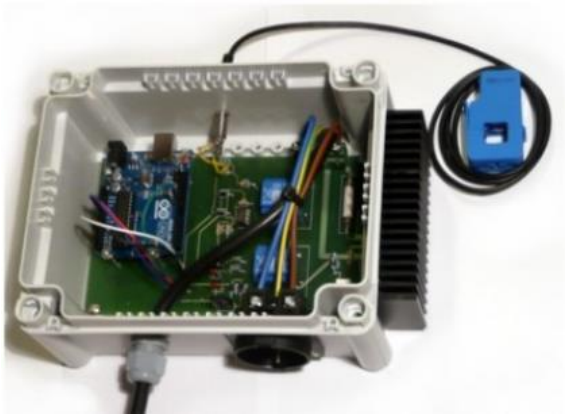

Figura 3 - Inibidor de Injeção na Rede

Este dispositivo disponibilizado pela "http://www.solarshop.pt" analisa a energia da casa e caso esteja a ser produzida mais energia do que a necessária, ele faz o desvio desse excesso de energia para um dispositivo de consumo. É um excelente método, pois garante que não exista energia desperdiçada por envio desta para a rede, mas a sua desvantagem é que só pode ser utilizado com cargas puramente resistivas, como o caso de alguns aquecedores, ou cilindros de água quente sanitárias, isto limita bastante a sua utilização, principalmente no verão onde pelas melhores condições atmosféricas e horas de sol existe mais produção de energia. (Solar Shop, s.d.)

3. METODOLOGIA E ANÁLISE DE REQUISITOS

Neste capítulo é descrita a metodologia selecionada, o desenvolvimento da aplicação e a análise de requisitos na sua elaboração. Ainda é feita uma pequena análise de funcionamento aos principais sistemas com que o presente trabalho vai interagir.

3.1. Metodologia

Este projeto está a ser desenvolvido escolhendo o método de desenvolvimento ágil, isto irá fazer com que todo o processo seja desenvolvido por etapas. Ao ser dividido o trabalho, é possível definir uma ordem de trabalho especifica para que no final todo o conjunto funcione tal como esperado.

Devido a este sistema ter de interagir com outros sistemas já existentes vai ser necessário perceber como todo o sistema energético de uma habitação funciona, perceber onde é possível melhorar, analisar os dados obtidos e projetar uma solução que seja possível ser adaptada a outras habitações em situações semelhantes.

3.2. Planeamento

Visto que o projeto interage com diversos sistemas já existentes é necessária uma pequena análise a esses mesmos sistemas e entender o seu funcionamento para depois ser possível pensar numa solução viável. A primeira análise será ao funcionamento do sistema de energia solar numa habitação, passando depois a análise de uma solução para consumo dessa energia. Depois de obtidos alguns dados é que será possível decidir se é viável implementar um destes tipos de solução.

3.3. Análise da rede elétrica de uma habitação

Hoje em dia todas as habitações possuem um sistema elétrico por toda a habitação. Cada vez mais, existem dispositivos eletrónicos responsáveis por consumir essa energia sejam eles eletrodomésticos, sistemas de som, sistemas de imagem, redes de dados, todos eles consomem energia. O Sistema elétrico da habitação é alimentado pela rede elétrica nacional, cada habitação possui uma ligação à rede. À entrada da rede existe um contador de energia responsável por contabilizar a energia gasta na habitação, e um quadro elétrico com disjuntores responsáveis por distribuir a energia pelas diferentes áreas e necessidades da casa. Entre outras seguranças, os disjuntores têm um grande papel de funcionamento, pois estão responsáveis por desligar a conexão ao seu circuito caso exista sobrecarga ou curto-circuito. No caso de um sistema solar, esse sistema tem o seu próprio disjuntor no quadro elétrico para maior segurança.

Analisando os consumos das habitações existe um eletrodoméstico que é comum a maior parte delas, hoje em dia a maior parte das habitações possui uma arca frigorífica. Uma arca frigorífica é um eletrodoméstico com um consumo considerável, que necessita de estar sempre conectada à rede, mas nem sempre está a consumir energia. Partindo deste princípio será um bom caso de estudo.

3.4. Análise do sistema solar fotovoltaico

O sistema solar da tipologia autoconsumo é constituído por duas partes essenciais, sendo elas os painéis que produzem a energia e o inversor.

Os painéis podem estar numa posição fixa ou numa estrutura que segue o sol. Quando estão numa estrutura de seguimento solar, estes conseguem produzir cerca de 30% mais energia do que se estiverem montados numa posição fixa, pois o seguidor orienta-os, ao longo do dia, para que o sol incida neles na sua posição ótima. Embora seja muito vantajoso um seguidor solar eleva o custo de um sistema destes. Os painéis são ligados diretamente ao inversor.

O inversor é a peça responsável por transformar a energia produzida pelos painéis em energia compatível com a rede elétrica nacional e injetar toda a energia que eles estão a produzir no momento diretamente na rede da habitação.

Estes sistemas estão normalmente dimensionados para cobrir o consumo da habitação em que estão instalados, mas este consumo não é fixo durante todo o tempo, logo é difícil de conseguir não desperdiçar alguma dessa energia por não ser consumida no momento em que está disponível.

Este sistema é o mais usual hoje em dia, é o tipo de sistema de energia solar mais fiável e mais económico, não requer nenhum tipo de manutenção especial, o inversor é facilmente instalado sem ocupar espaço dedicado na habitação, unicamente é preciso fazer uma ligação ao quadro elétrico nela existente.

3.5. Análise de uma arca frigorífica

Uma arca frigorífica, tipicamente é uma caixa fechada para conservar os alimentos congelados a uma temperatura negativa. Para conseguir manter essa temperatura ela possui um sistema que absorve o calor do lado interior e liberta-o do lado exterior. Para o sistema funcionar, existe um termóstato responsável por monitorizar a temperatura interior e no caso de estar acima do parâmetro definido, ele faz funcionar todo o sistema responsável por arrefecer o interior. As arcas frigoríficas são termicamente isoladas para conseguir manter o interior frio e assim não precisarem estar sempre a funcionar.

O sistema que arrefece o interior é constituído por um compressor com partes mecânicas, isto faz com que seja necessária alguma precaução no que diz respeito a tempos de iniciar e parar o seu funcionamento.

Com esta ideia será necessário a criação de um módulo eletrónico de recolha de dados com o objetivo de analisar melhor este eletrodoméstico e verificar se será possível tirar algum proveito dele.

4. TECNOLOGIAS UTILIZADAS

Neste capítulo serão apresentadas as tecnologias utilizadas para o desenvolvimento da aplicação, assim como algumas características das mesmas.

4.1. Arduíno

O Arduíno é uma das plataformas mais conhecidas no mundo da programação para eletrónica de hardware. Tem uma base fácil de programar através de uma interface no computador onde por base é utilizado a linguagem de programação C com extensões que permitem controlar o hardware ligado fisicamente à placa. Existem muitas versões do Arduíno, cada uma tem especificações diferentes. A usada neste projeto é a plataforma Arduíno Nano (Figura 4) por ser compacta e conseguir satisfazer todas as necessidades a que será submetida.

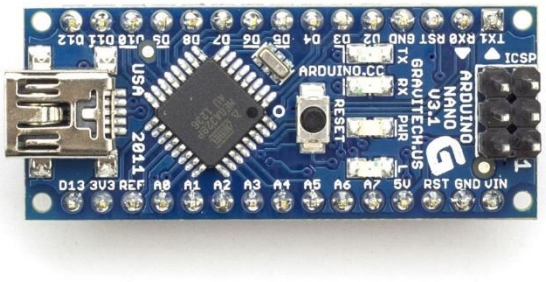

Figura 4 - Arduíno Nano

O Arduíno Nano é composto por uma pequena placa de circuito impresso, compatível com as breadboards prototipagem eletrônica. O principal componente da placa é o seu microcontrolador Atmega 328P que gere os seus pinos de comunicação com ouros dispositivos e consegue ser programado para executar um trabalho definido pelo utilizador.

Nesta placa podemos encontrar todos os componentes eletrónicos necessários para se poder ligar numa vulgar porta USB de um computador de maneira a ser programado e ainda uma

comunicação de dados com o computador, possui também 14 pinos de ligações digitais e ainda 8 analógicos, todos eles trabalham com voltagens logicas (tipicamente 0-5V). Por ser uma base bastante explorada, é possível ligar a ele os mais diversos sensores e atuadores. Com facilidade encontram-se bibliotecas online para os dispositivos com que iremos trabalhar. (Arduino cc, s.d.)

4.2. ESP8266

O ESP8266 (Figura 5) é um pequeno modulo eletrónico com um pequeno microcontrolador que tem a capacidade de comunicar por redes Wireless.

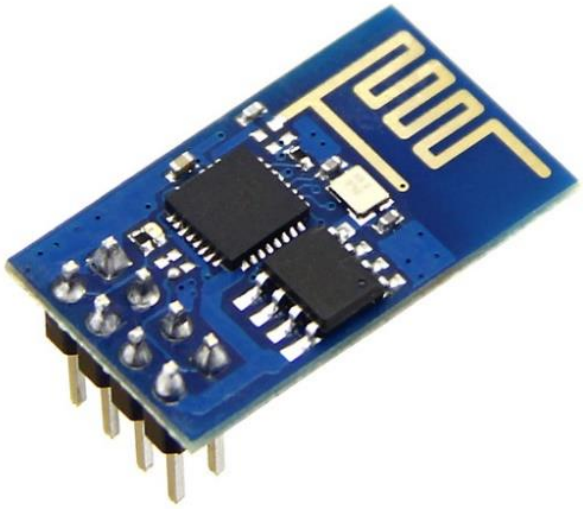

Figura 5 - ESP8266

O módulo consegue conectar-se a redes Wireless, fazer comunicações TCP/IP e para o comandar são utilizados comandos AT. Os comandos AT definem uma linguagem de comunicação e são conhecidos por serem utilizados em diferentes dispositivos como telemóveis, módulos Bluetooth e outros módulos de comunicação. Este modulo tem a desvantagem de não funcionar na voltagem lógica padrão (5V) mas funciona a 3.3V, mesmo assim é possível ligar ao Arduíno e conseguir com que o Arduíno comunique através dele com a rede, seja ela interna ou externa (internet).

A sua aplicabilidade é muita, hoje em dia com a “internet das coisas”, é muito comum encontrar pequenos dispositivos com capacidade de comunicar com a internet.

Principais Comandos AT		
Comando	Descrição	Resposta
AT	Testa a comunicação e funcionamento do módulo	"OK" caso esteja em correto funcionamento
AT+RST	Fazer reset ao módulo	Reinicia e fica pronto quando responder "ready"
AT+CWMODE=1	Definir modo de funcionamento, neste caso em modo cliente Wireless(1)	"OK" ou "no change" caso não proceda a alteração
AT+CWLAP	Procurar e listar redes wireless	Lista de redes wireless
AT+CWJAP="SSID","Password"	Ligar a uma Rede Wireless	Caso obtenha sucesso "OK" senão "ERROR"
AT+CIFSR	Verificar o IP	IP atribuído ao módulo
AT+CIPSTART=	Fazer uma ligação TCP/UDP	"OK" caso tenha sucesso
AT+CIPSEND=<tamanho>	Envio de dados	">" e aguarda pela string de dados a enviar
AT+CIPCLOSE	Terminar a conexão	"OK"

A resposta do módulo varia consoante o firmware que esteja instalado. No comando "AT+CWMODE=1" podem ser definidos mais modos, mas o modo utilizado neste trabalho é o modo 1 onde o ESP8266 trabalha em modo cliente de uma rede Wireless.

4.3. ThingSpeak

O ThingSpeak é uma ferramenta online gratuita muito versátil para todo o tipo de aplicações que necessitem de um serviço na internet que consiga armazenar dados, enviar comandos ou mesmo tratar a informação. Neste projeto unicamente será utilizado uma parte do ThingSpeak que é conhecido como "canal". Um canal do ThingSpeak é um sítio na web que está sempre disponível para receber os dados de pequenas aplicações e guardar numa base de dados. Para ser utilizado o canal é preciso conectar ao servidor do ThingSpeak, e através de uma chave única fornecida (API) pelo ThingSpeak enviar os dados para o servidor. O servidor do ThingSpeak está configurado para receber os dados utilizando o método HTTP GET ou POST para o endereço

https://api.thingspeak.com/update. Nesse endereço tem de ser incluído os parâmetros necessários como a chave (API) e os dados a enviar, ficando algo como:

https://api.thingspeak.com/update?api_key=XXXXXXXXXXXXXXXX&field1=30

Neste caso o servidor do ThingSpeak vai guardar no canal com a respetiva chave, no campo 1 o valor 30, e no website é apresentado o gráfico com o valor (Figura 6).

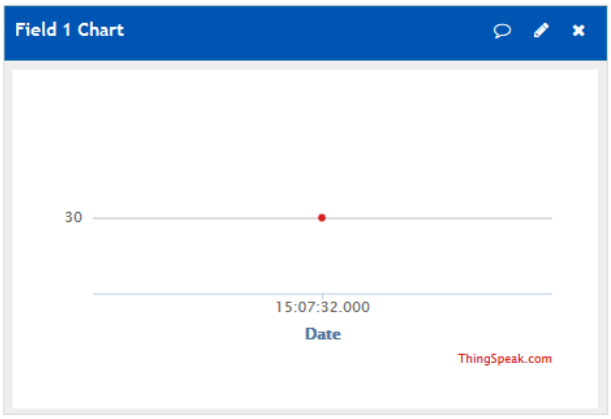

Figura 6 - Gráfico ThingSpeak

É uma ferramenta muito útil para este caso em específico, pois basta uma conexão à internet e é possível fazer uma recolha de dados com a vantagem de poder seguir os resultados em tempo real.

4.4. Módulos Comunicação RF

Os módulos de comunicação por rádio frequência (Figura 7) são pequenos módulos eletrónicos que funcionam como uma ligação virtual, com a desvantagem de ser apenas num sentido, pois existe o módulo que é capaz de fazer a transmissão e existe outro módulo capaz de fazer a receção.

Figura 7 - Módulos RF 433MHz

Estes módulos permitem interligar dispositivos a curtas distâncias de uma forma fácil, barata e não precisam de nenhum tipo de infraestrutura especial.

4.5. Sensor Temperatura DS18B20

Neste projeto para a leituras de temperaturas foi escolhido o sensor DS18B20 (Figura 8). Este sensor foi escolhido pelas suas características, tendo em atenção às necessidades deste projeto. O sensor consegue ler temperaturas desde -55°C ate +125°C o que garante um grande intervalo de segurança para as leituras utilizadas. Uma grande vantagem em relação aos sensores de temperatura mais conhecidos é possuir o seu próprio controlador interno incorporado, isto permite ler do sensor valores com maior resolução e precisão. A única desvantagem dele é ser necessário utilizar um protocolo de comunicação específico conhecido como “OneWire”, pois o sensor utiliza um único pino para fazer a sua comunicação bidirecional.

Figura 8 - Sensor temperatura DS18B20

Ele tem muitas outras vantagens devido a possuir o seu próprio controlador, uma das quais é ser possível ligar vários sensores todos na mesma linha de comunicação, pois cada um tem o seu endereço único. Este sensor tem uma resolução que permite ser programada, caso se precise de mais detalhe ou menos detalhe nos dados. O conversor interno vem programado para o nível máximo de detalhe (12bits), isto origina um tempo de espera de 750ms desde que lhe damos a ordem de leitura e fica disponível para leitura, mas caso seja necessária uma resposta mais rápida e menos precisa programando o sensor para o nível de detalhe mínimo (9bits) onde o seu tempo de espera reduz para apenas 94ms (alldatasheet.com, s.d.).

5. DESENVOLVIMENTO

Neste capítulo são apresentados os desenvolvimentos dos módulos de trabalho.

5.1. Módulo para obtenção de resultados

Para verificar a viabilidade do projeto foi necessário projetar um módulo para recolher dados do funcionamento real de uma arca congeladora. A base programável deste módulo foi um Arduíno Nano. Para a leitura de dados foi escolhido o Sensor DS18B20 devido às suas características como explicado anteriormente. A principal questão era o armazenamento dos dados para posterior análise. Após alguma pesquisa, o serviço disponibilizado pelo website "ThingSpeak" apresentou ser ideal para o propósito, mas criou a necessidade de implementar um módulo de comunicação com a internet. O módulo ESP8266 foi o escolhido para este propósito.
Começou-se por o interligar o Arduíno com o sensor de temperatura, visível na Figura 9. O sensor de temperatura partilha a alimentação de 5V (VCC) e o GND com o Arduíno e o pino de dados liga diretamente com o Arduíno, mas por necessidade do protocolo de comunicação, é necessário implementar uma resistência de *pull up* com o valor de 4.7k Ohm(R3) interligada entre o VCC e a linha de dados. Por segurança foi adicionado um condensador (C2) para maior estabilidade da alimentação do circuito.

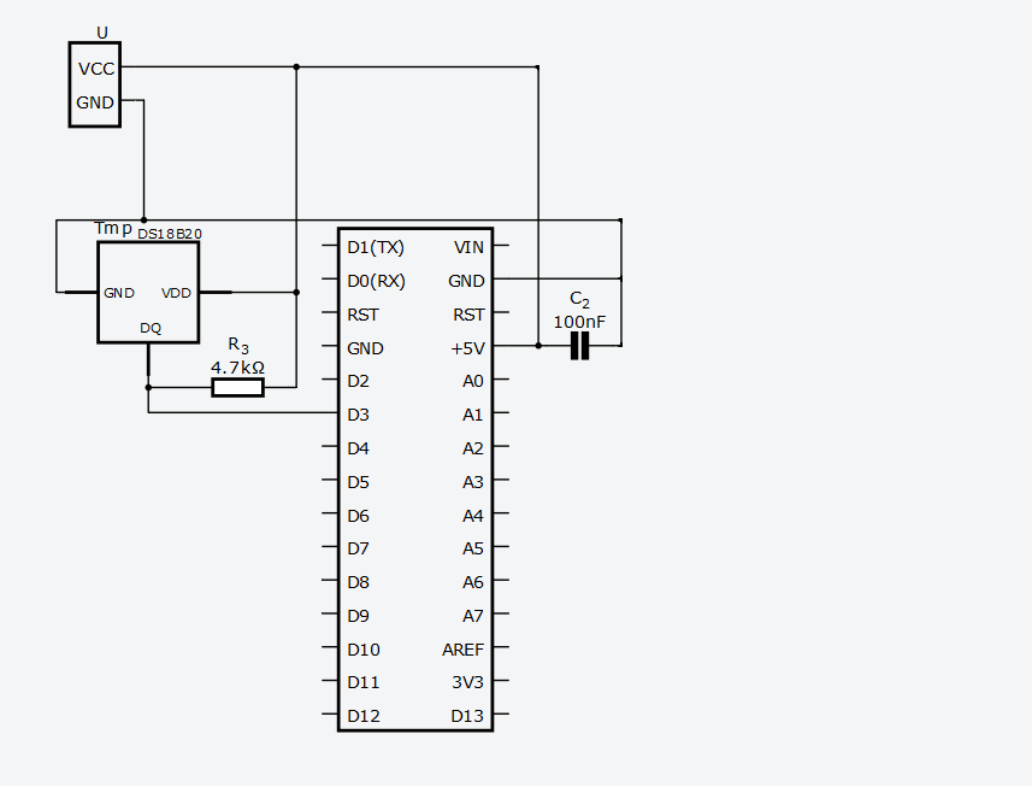

Figura 9 - Arduíno com o sensor de temperatura

Após este passo concluído, foi interligado o módulo ESP8266 com o Arduíno (Figura 10). O módulo ESP8266 funciona numa voltagem diferente do Arduíno. Enquanto que o Arduíno funciona a 5V o ESP8266 funciona a 3.3V. Como a voltagem do módulo é inferior a voltagem do Arduíno é possível utilizar um regulador linear para reduzir a voltagem para o módulo ESP8266. O componente AMS1117 cumpre na perfeição essa função. Para uma melhor estabilização da tensão de saída de 3.3v foi adicionado um condensador (C1). Para comunicar entre o Arduíno e o módulo ESP8266 são utilizadas duas ligações (Tx, Rx), mas como as alimentações são diferentes, os níveis lógicos também diferem, é preciso analisar como fazer a ligação entre os dispositivos para não existir problemas de queimar os componentes.

No caso de o Tx do ESP8266 comunicar com o nível lógico de 3.3V para o Rx do Arduíno não existe problema, pois o Arduíno reconhece os 3.3v como nível lógico alto e funciona perfeitamente. Analisando o caso do Tx do Arduíno comunicar no nível lógico de 5V para o Rx do ESP8266 que funciona a 3.3V existe um problema pois o módulo ESP8266 pode

queimar devido a receber voltagem superior. Para resolver este problema foi usado duas resistências (R1 e R2) que causassem uma queda de tensão de 5V para 3.3V, ficando assim adequada a tensão que o ESP8266 vai receber. Como o Arduíno Nano só tem disponível uma porta Serial (Rx, Tx) que é utilizada para comunicação com o computador, foi utilizada uma biblioteca que consegue emular por software uma outra porta serie em dois pinos a escolha do programador, assim sendo foi escolhido os pinos 8 e 9 para a ligação ao ESP8266. A resistência R4 é opcional, mas aconselha-se o seu uso para o bom funcionamento do ESP8266

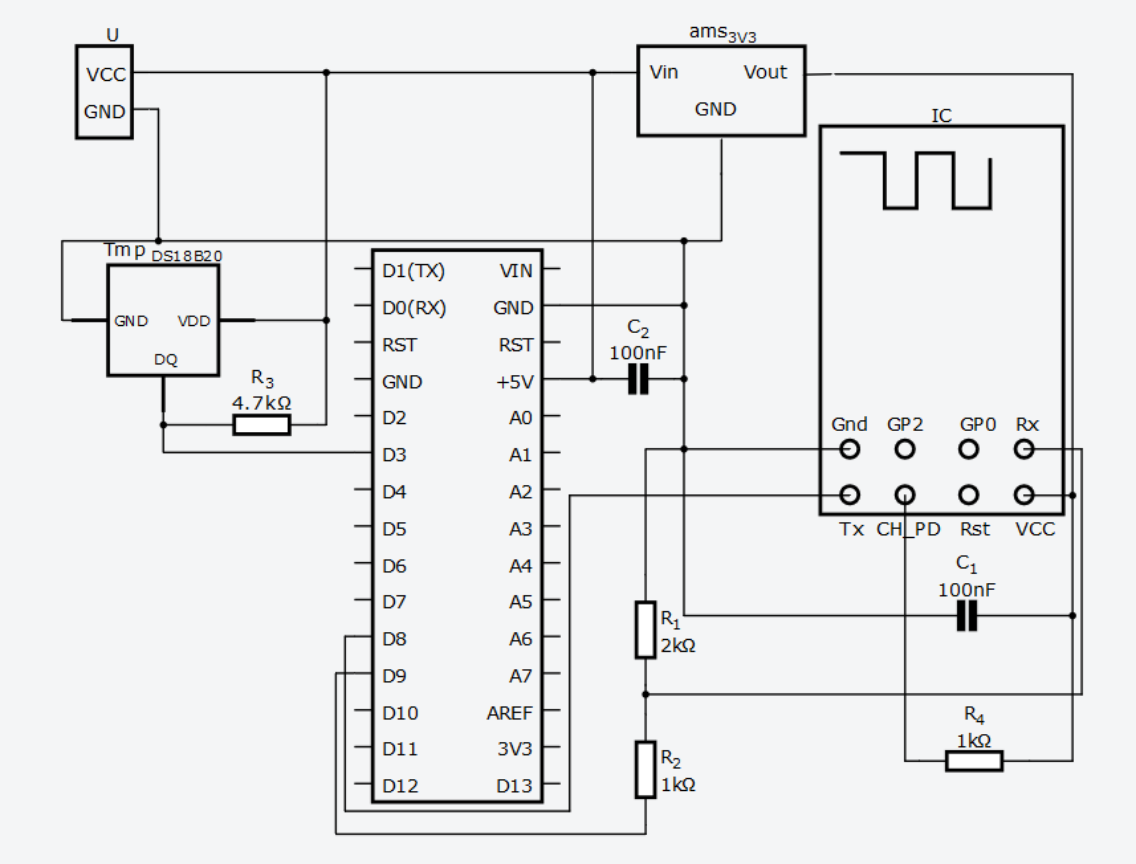

Figura 10 - Módulo de obtenção de dados completo

Antes de passar à programação, é necessário fazer o algoritmo de funcionamento deste módulo, para assim se seguir com a programação. O algoritmo para este módulo é o seguinte:

Função *setup()*

1. Iniciar a porta serie de comunicação com o PC

2. Iniciar a porta serie para comunicação com o ESP8266
3. Estabelecer a ligação WiFi

Função *loop()*

1. Se passou o tempo definido
   1.1.Ler e enviar dados

Função conectarWifi()

1. Enviar o comando “AT” ao ESP8266
2. Se encontrar a resposta “OK”
   2.1. Reiniciar ESP8266
   2.2. Configurar em modo cliente
   2.3. Configurar a rede a ligar
   2.4. Configurar para ligações múltiplas

Função enviarTemperatura()

1. Ler a temperatura
2. Fazer a ligação ao ThingSpeak
3. Se a resposta do ESP8266 for “Error”
   3.1. Conectar de novo a rede Wi-Fi
   3.2. Terminar esta função
4. Criar a string de dados para o ThingSpeak
5. Enviar o comando de envio de dados
6. Se na resposta encontrar o caracter “>”
   6.1. Enviar os dados
7. Fechar a conexão
8. Se na resposta não encontrar “OK”
   8.1.Conectar de novo a rede Wi-Fi

Função getTemp()

1. Fazer reset na linha de dados
2. Enviar a ordem de ignorar o endereço
3. Enviar a ordem de conversão para o sensor
4. Esperar que o sensor converta a temperatura
5. Fazer reset na linha de dados
6. Enviar a ordem de ignorar o endereço
7. Enviar a ordem para o sensor enviar os dados
8. Ler os dados para o array de dados
9. Fazer reset na linha de dados
10. Devolver a temperatura lida em graus ºC

Passando ao código de Arduíno, o primeiro passo foi o sensor de temperatura.

Como o sensor de temperatura utiliza um protocolo de comunicação vamos utilizar a biblioteca “OneWire”, mas esta biblioteca é uma derivação da biblioteca original “Wire”

```
#include <Wire.h>
#include <OneWire.h>

#define SensorPin 3
OneWire ds(SensorPin);
```

Após incluídas as bibliotecas foi definido o pino onde está ligado o sensor de temperatura e após isso iniciado o mesmo.

Para ler a temperatura do sensor é necessária comunicar com ele de forma ordenada, logo foi utilizada uma função. Nessa função primeiramente é necessário enviar a ordem de ignorar o endereço do sensor por existir só um, enviar a ordem para converter a temperatura e ficar disponível no seu registo interno, após isso, como o sensor na máxima precisão demora 750ms a converter a temperatura, faz-se uma pausa antes de pedir ao sensor que envie a sua temperatura. Como a temperatura vem em dois bytes é necessário converter para um inteiro e depois para um float.

```
float getTemp() {
  byte data[12];
  ds.reset();
  ds.write(0xCC);
  ds.write(0x44);
  delay(760);
  ds.reset();
  ds.write(0xCC);
  ds.write(0xBE);
  for (int i = 0; i < 9; i++) {
    data[i] = ds.read();
  }
  ds.reset();
  return ((float)(*((int*)data))/16.0);
}
```

A função faz todo esse trabalho, sendo só preciso chamar a função para que ela retorne o valor real da temperatura em graus centígrados.

Para utilizar o módulo ESP8266, é necessário incluir a biblioteca “SoftwareSerial” e definir quais os pinos que são utilizados como Rx e Tx

```
#include <SoftwareSerial.h>
SoftwareSerial ESP8266(9, 8); //RX, TX para o ESP8266
```

Após isso, na função de “Setup” do Arduíno é necessário iniciar a porta série emulada.

```
ESP8266.begin(9600); //velocidade de comunicacao por porta serie com o ESP8266 (serie emulada)
```

Estando estes passos feitos é possível comunicar com o ESP8266 como se estivesse ligado a uma porta serie nativa. Para facilitar a comunicação com o ESP8266 usa-se uma função, estando esta responsável por lhe enviar os comandos e aguardar pela sua resposta.

```
String ComandoAT(String comando, const int timeout)
{
  String resposta = ""; //variavel para armazenar a resposta do ESP
  ESP8266.println(comando); //Envia os dados para o ESP para executar o comando
  long int time = millis(); //variaval para armazenar os millis atuais
  while ( (time + timeout) > millis()) {
    while (ESP8266.available()) //utilizado para armazenar o resultado enviado pelo ESP
    {
      char c = ESP8266.read();
      resposta += c;
    }
  }
  return resposta;
}
```

Como o ESP8266 é um simples módulo de comunicação, é necessário iniciá-lo, e especificar o que ele deve fazer, para isso existe a seguinte função, responsável por o configurar como cliente wireless e ligar à rede.

```
void conectarWiFi()
{
  ESP8266.println("AT");
  delay(10);
  if (ESP8266.find("OK"))
  {
    ComandoAT("AT+CWMODE=1", 1000); //configura acesspoint
    String cmd = "AT+CWJAP=\"";
    cmd += SSID;
    cmd += "\",\"";
    cmd += PASS;
    cmd += "\"";
    ComandoAT(cmd, 10000);
    ComandoAT("AT+CIFSR", 1000);
    ComandoAT("AT+CIPMUX=1", 1000);
    Serial.println("\n Ligado a Rede");
  }

}
```

O nome da Rede é a respetiva password estão definidos no início do programa

```
#define SSID "                    " //nome da rede de WIFI a ligar
#define PASS "        " //password da rede WIFI
```

O ESP8266 é um módulo muito versátil, existem muitas versões do seu software, por isso dependendo da versão poderá ser necessário configurações diferentes.

No ThingSpeak foi criado um canal, tendo sido gerada uma chave única que permite o envio dos dados recolhidos pelo módulo. Para além dessa API também é necessário o IP do servidor do ThingSpeak.

```
#define IP "184.106.153.149" //ThingSpeak IP
#define APIKey "WTBIB1[illegible]1HW" //ThingSpeak API Key
```

Após isto existe uma função para executar todo o processo de enviar os dados para o Thingspeak.

```
void updateTS() {
  String GET = "GET /update?key="; //URL para fazer upload para thingspeak
  GET += APIKey;
  String cmd = "AT+CIPSTART=4,\"TCP\",\""; //Iniciar uma ligacao TCP
  cmd += IP;
  cmd += "\",80";
  ComandoAT(cmd, 1);
  cmd = "";
  delay(1000);
  temperatura = getTemp();
  if ( ESP8266.find( "Error" )) {
    Serial.println("\n Falhou, vai recomeçar");
    connectWiFi();
    return;
  }
  cmd = GET + "&field1=" + String(temperatura) + "\r\n";
  ESP8266.print( "AT+CIPSEND=4," );
  ESP8266.println(cmd.length());
  delay(1000);
  if (ESP8266.find( ">" )) {
    ESP8266.print(cmd); //envia dados para o thingspeak
    delay(100);
  }
  else {
    ESP8266.println( "AT+CIPCLOSE=4"); //fechar ligacao TCP
    delay(1000);
  }
  if ( ESP8266.find("OK") ) {
    Serial.println("Conexao fechada");
  }
  else {
    Serial.println( "\n Falhou, vai recomeçar" );
    connectWiFi();
  }
}
```

Esta função começa por fazer uma ligação TCP ao servidor do ThingSpeak, recolhe a temperatura do sensor, envia a string de dados para o ThingSpeak, e por fim termina a ligação ao servidor.

Após, este pequeno trabalho foi possível fazer uma análise a dados reais.

5.2. Análise de resultados obtidos

Após o módulo anterior estar concluído, foi possível recolher dados para uma análise mais detalhada. A Arca frigorífica utilizada é de classe energética A++, o que garante uma enorme eficiência. O consumo é de apenas 80w enquanto está em trabalho.

Durante a recolha obtiveram-se os dados da Figura 11.

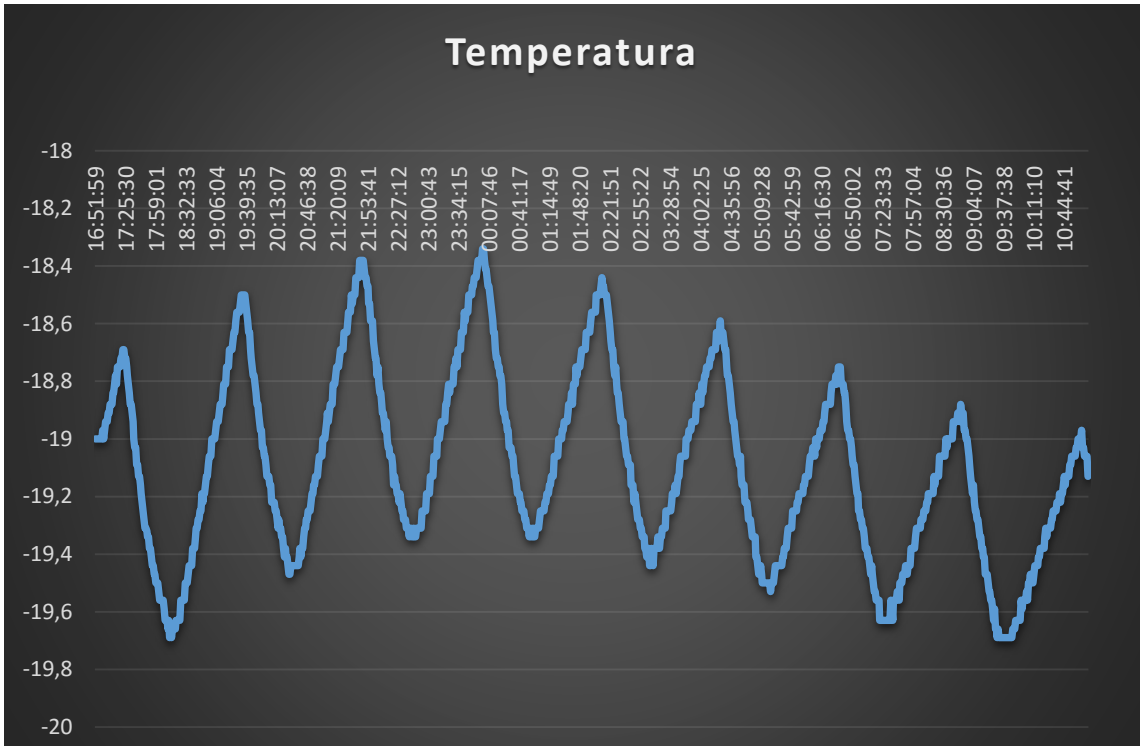

Figura 11 - Dados obtidos

Após uma análise detalhada aos dados, a arca frigorífica tem um período de trabalho de cerca de 52 minutos, e um período de repouso de cerca de 80 minutos. Ao longo de um dia (24h), a arca frigorífica estará em repouso aproximadamente 14:33 horas e em trabalho 9:27 horas.

5.3. Módulo de monitorização de consumo de energia

Como falado anteriormente, o projeto final consiste em dois módulos que comunicam entre si. O módulo principal é responsável por analisar o consumo da habitação e a produção de energia solar. Este módulo, tendo em conta os parâmetros previamente configurados, vai decidir se envia ou não ordens para o segundo módulo trabalhar em modo de energia renovável.

Para este módulo funcionar corretamente tem de ficar instalado no quadro geral elétrico da habitação.

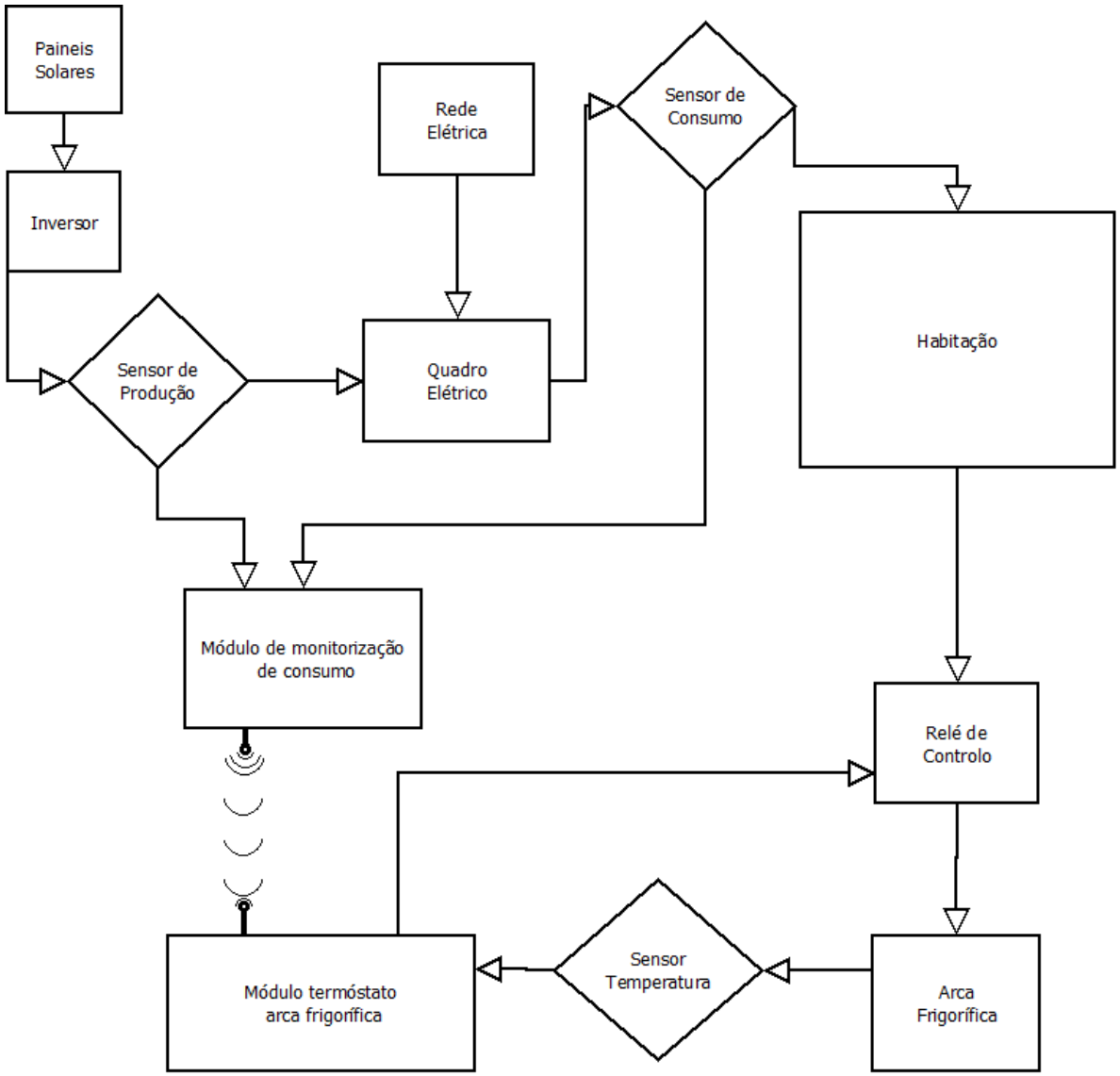

Figura 12 - Diagrama de Blocos do Sistema

Na Figura 12 está o diagrama de blocos do sistema. Para este caso, em específico, seria possível utilizar apenas um sensor de corrente bidirecional, mas isso trazia algumas desvantagens para possíveis melhorias futuras do sistema.

Como não foi possível implementar esta parte do projeto numa habitação real, para a elaboração do presente trabalho os sensores foram substituídos por potenciómetros, a fim de ser possível fazer testes a todo o sistema e comprovar o seu funcionamento.

No hardware foi utilizado um Arduíno nano com o microcontrolador atmega328P (Figura 13). Ligado a ele estão os dois potenciómetros (R3 e R4), que substituem os sensores de corrente para testes. Está ainda inserido um condensador (C1) para maior estabilidade da alimentação do circuito.

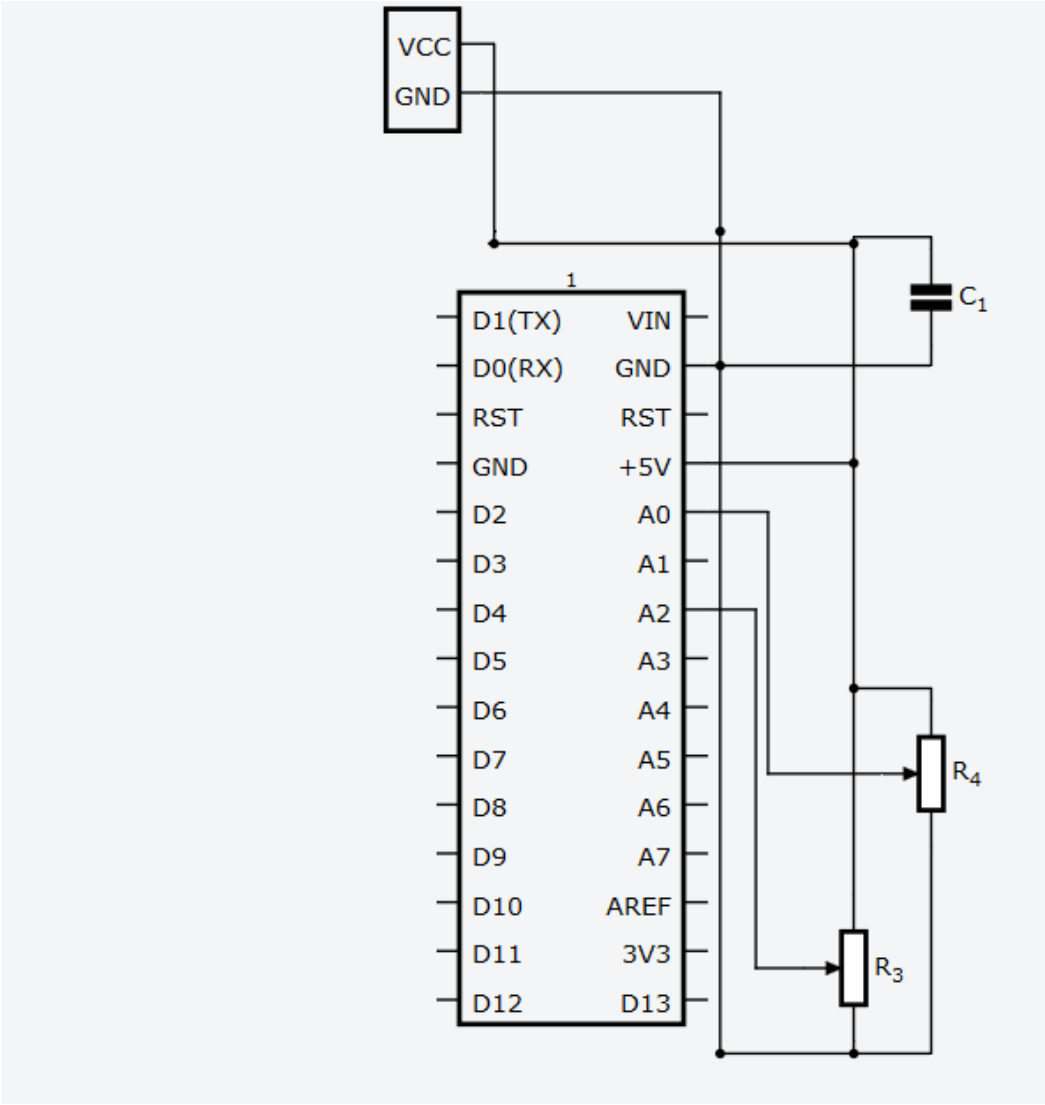

Figura 13 – Ligações do Módulo de monitorização de consumo

Em Seguida (Figura 14), foram incluídos 2 led's de estado (D1, D2) cada um com a sua resistência (R1, R2).

Um led é vermelho e fica ativo caso o consumo seja superior à produção.

Caso exista uma produção superior ao consumo e a margem seja de acordo com o definido na programação o led vermelho desativa e ativa o led Verde.

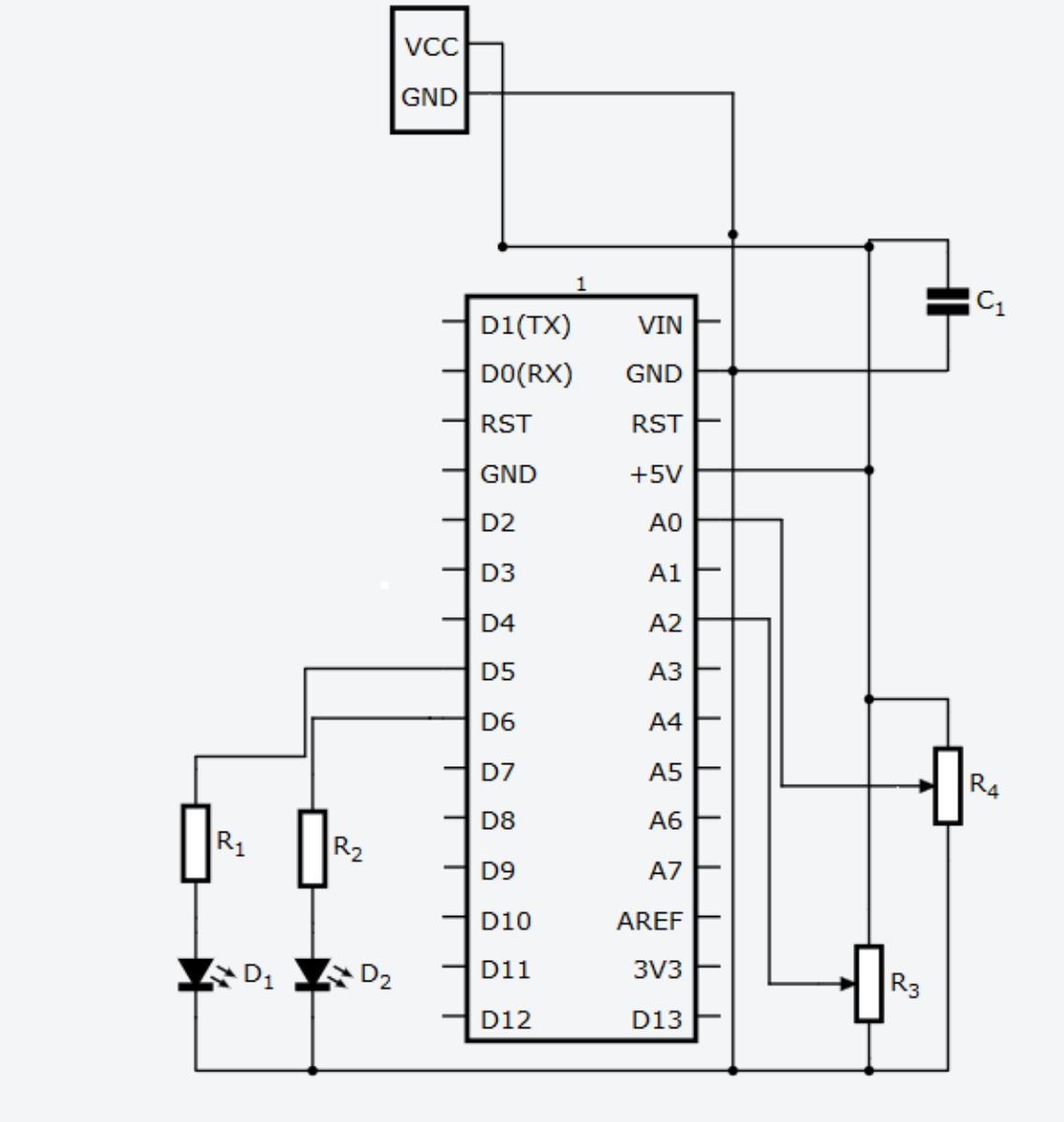

Figura 14 – Ligações do Módulo de monitorização de consumo

Após isto, só falta implementar o transmissor RF 433MHz para poder enviar ordens para o outro módulo (Figura 15). O módulo transmissor precisa de alimentação e uma linha de dados.

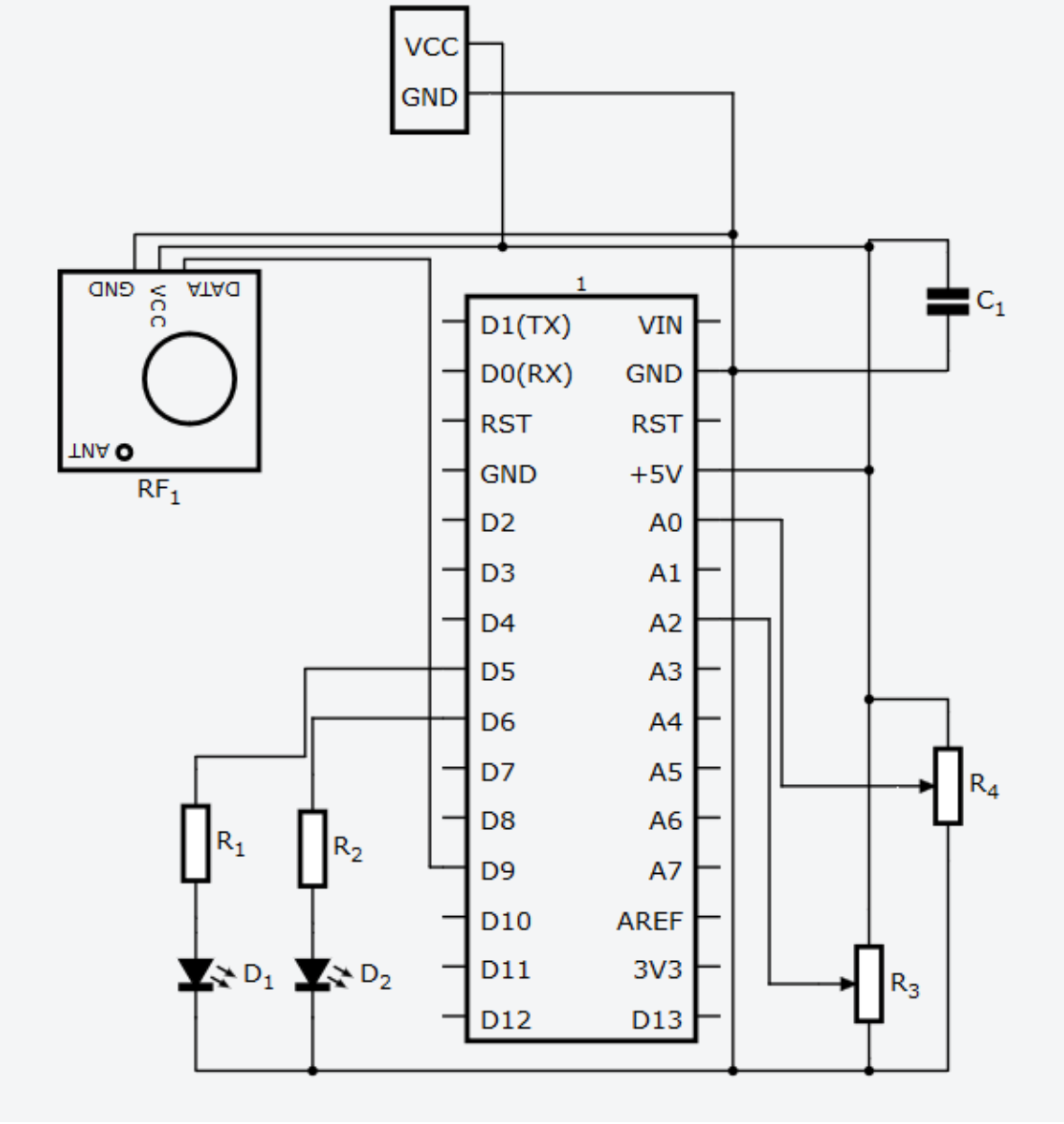

Figura 15 - Módulo de monitorização de consumo completo

Na criação do algoritmo obteve-se o seguinte:

Função *setup()*

1. Configurar o pino de transmissão de dados
2. Configurar o modo de envio
3. Configurar a velocidade de envio

4. Iniciar a porta serie de comunicação com o PC
5. Configurar o pino do led verde como OUTPUT
6. Configurar o pino do led vermelho como OUTPUT

Função *loop()*

1. Se passou o tempo definido
   1.1. Atualizar os dados

Função *funcaoAtualizar()*

1. Se a diferença absoluta entre o valor do consumo e o valor da produção for superior ou igual ao valor de margem
   1.1. Se o consumo for maior ou igual à produção
      1.1.1. Desliga o *LedVerde*
      1.1.2. Se ainda não atingiu o tempo mínimo de espera
         1.1.2.1.Incrementa o contador
         1.1.2.2.Inverte o estado do pino do *LedVermelho*
      1.1.3. Senão
         1.1.3.1.Liga o LedVermelho
         1.1.3.2.Atualiza o contador com o valor 0
         1.1.3.3.Se estiver em Renovável
            1.1.3.3.1. Envia a ordem de Normal
            1.1.3.3.2. Atualiza o estado para Normal
   1.2.Senão
      1.2.1. Desliga o *LedVemelho*
      1.2.2. Se ainda não atingiu o tempo mínimo de espera
         1.2.2.1.Incrementa o contador
         1.2.2.2.Inverte o estado do pino do *LedVerde*
      1.2.3. Senão
         1.2.3.1.Liga o *LedVerde*
         1.2.3.2.Atualiza o contador com o valor 0
         1.2.3.3.Se estiver em Normal
            1.2.3.3.1. Envia a ordem de Renovável
            1.2.3.3.2. Atualiza o estado para Renovável

Função *renovavelON*()

1. Criar *array* com a mensagem “1#”
2. Enviar a mensagem
3. Esperar que a mensagem seja totalmente enviada.

Função *renovavelOFF*()

1. Criar *array* com a mensagem “0#”

2. Enviar a mensagem
3. Esperar que a mensagem seja totalmente enviada.

Na parte da programação foi necessário adicionar uma biblioteca para a transmissão de dados por RF. (VirtualWire Library, s.d.)

```
#include <VirtualWire.h>
```

Após a biblioteca inserida é necessário definir qual o pino onde o emissor esta ligado.

```
#define transmit_pin 9 //pino transmitir
```

Estando já definido é possível iniciar a biblioteca na função “setup”

```
vw_set_tx_pin(transmit_pin);
vw_set_ptt_inverted(true); //configurar a polaridade da comunicacao
vw_setup(2000); //bits por segundo
```

Estando a parte do transmissor definido resta definir as mensagens que ele vai enviar. Para isso, foram criadas as duas funções.

```
void renovavelON(){
  char msg[2] = {'1','#'};
  vw_send((uint8_t *)msg, 2);
  vw_wait_tx();
}

void renovavelOFF(){
  char msg[2] = {'0','#'};
  vw_send((uint8_t *)msg, 2);
  vw_wait_tx();
}
```

Sempre que seja necessário enviar dados para o segundo módulo só é preciso invocar a função, ela trata de fazer a mensagem, enviar a mensagem através de um interrupt definido na biblioteca e esperar que a mensagem seja enviada antes de libertar o microcontrolador para a função seguinte.

Para os led’s de estado foi necessário definir em que pino do Arduíno estes estão ligados.

```
#define LedVerde 5 //pino led verde
#define LedVermelho 6 //pino led vermelho
```

Após estarem definidos os pinos de ligação, é necessário configurá-los na função Setup como pinos de saída.

```
pinMode(LedVerde, OUTPUT);
pinMode(LedVermelho, OUTPUT);
```

Na função Setup, ainda foi iniciada a porta Serial de comunicação com o computador para ser possível observar a execução.

```
Serial.begin(115200);
```

Para as configurações iniciais foram definidas constantes no início do programa. Essas constantes têm de ser previamente definidas de acordo com a instalação onde seja instalado.

```
const int segEsperaON = 10; //segundos espera antes da ordem de ligar
const int segEsperaOFF = 10; //segundos espera antes da ordem de desligar
const int margemToler = 20; //diferenca entre sensores para tomar decisao
const long interval = 1000; //intervalo de (1000ms) para o millis
```

Ainda para o funcionamento de todas as funções do programa foram iniciadas algumas variáveis globais.

```
unsigned long previousMillis = 0; //valor da funçao millis
int sensorConsumoCasa = 0; // valor de consumo da casa
int sensorProducaoSolar = 0; //valor de producao de energia
int segON=0; //intervalo de estado ligado
int segOFF=0; //intervalo de estado desligado
int aux=0; // variavel auxilixar para calculo
bool ligado=LOW; //variavel de estado
```

Para o programa executar em intervalos de tempo corretos, na função loop() do Arduíno existe um código para verificar se já passou o intervalo de tempo previamente definido utilizando a função millis().

```
void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    funcaoTimer();
  }

}
```

Depois de passar o tempo correto invoca a função onde está presente todo o código de decisão do módulo. A "funcaoAtualizar()" começa por ler os dados dos sensores.

```
void funcaoAtualizar(){
  sensorConsumoCasa = analogRead(Sensor1);
  sensorProducaoSolar = analogRead(Sensor2);
  aux=abs(sensorConsumoCasa-sensorProducaoSolar);
```

Neste pedaço de código, o Arduíno vai ler cada sensor e atualiza a variável "aux" com o valor absoluto da diferença de sensores.

Na próxima fase, é necessário comparar os valores com os dados previamente definidos a fim de executar ordens.

A primeira condição "if" vai comparar se existe margem de tolerância para executar a comparação seguinte. Caso não exista termina a execução logo por aqui.

```
  if (aux >= margemToler) {
    if (sensorConsumoCasa >= sensorProducaoSolar) {

      digitalWrite(LedVerde, LOW);
      if (segOFF < segEsperaOFF) {
        segOFF++;
        digitalWrite(LedVermelho, digitalRead(LedVermelho) ^ 1);
      } else {
        digitalWrite(LedVermelho, HIGH);
        segON = 0;
        renovavelOFF();
        ligado = LOW;
      }
    } else
    {
      digitalWrite(LedVermelho, LOW);
      if (segON < segEsperaON) {
        segON++;
        digitalWrite(LedVerde, digitalRead(LedVerde) ^ 1);
      } else {
        digitalWrite(LedVerde, HIGH);
        segOFF = 0;
        renovavelON();
        ligado = HIGH;
      }
    }
  }
```

Caso exista margem ele vai seguir para a segunda condição "if", onde é verificado se a habitação se encontra a produzir ou a consumir.

Ainda depois desse processo vai existir uma outra condição “if” onde obriga a um tempo de espera, previamente definido, antes de enviar a ordem. Como é um sistema que está dependente da natureza e o funcionamento mecânico de uma arca frigorífica não está preparado para aguentar sucessivas paragens e inícios repentinos, um tempo de espera antes de enviar uma ordem ajuda a que não sejam enviados comandos caso passe uma nuvem ou outro fenómeno momentâneo.

5.4. Módulo termóstato arca frigorífica

Visto já estar o primeiro módulo em funcionamento e a enviar dados, é necessário um segundo módulo para os receber. Este vai interagir com a arca frigorífica para o consumo de energia no período certo.

O módulo possui uma sonda de temperatura para monitorização de temperatura interior da arca e uma saída que controla o funcionamento do compressor (Figura 16).

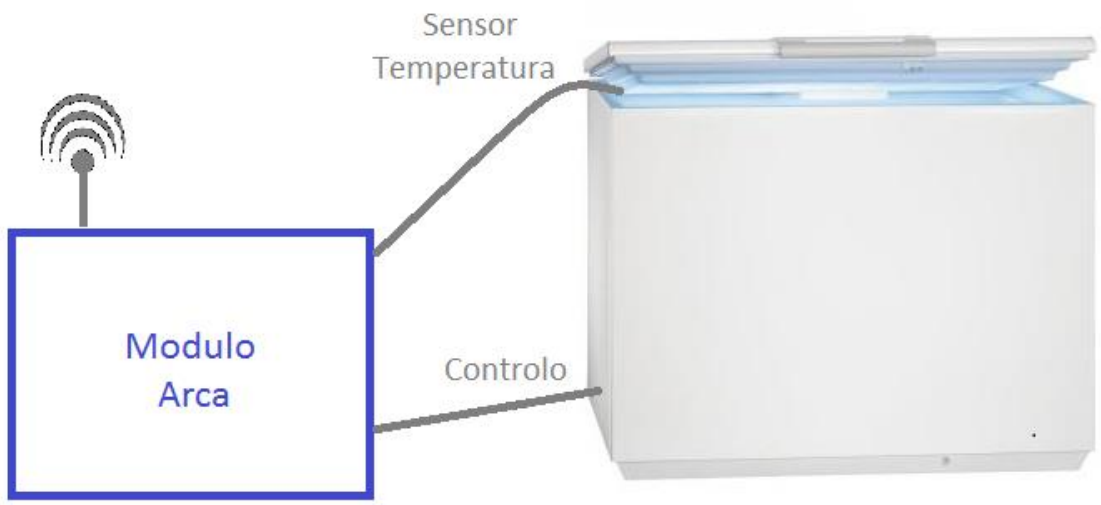

Figura 16 - Implementação do Módulo Arca Frigorifica

Desta forma é possível adaptar o módulo de duas maneiras possíveis. Uma da mareira será mais invasiva e é eliminar todo o controle de funcionamento da arca, e substituir por este

módulo. A outra menos invasiva, é unicamente passar o sensor de temperatura para dentro da arca através da borracha de vedação e fazer o controlo do motor através da ficha elétrica.

Na parte de hardware foi utilizado como no módulo anterior um Arduíno nano com o microcontrolador atmega328P, conectado a ele está o módulo de receção RF 433MHz, que apenas necessita de alimentação e um pino de comunicação (Figura 17).

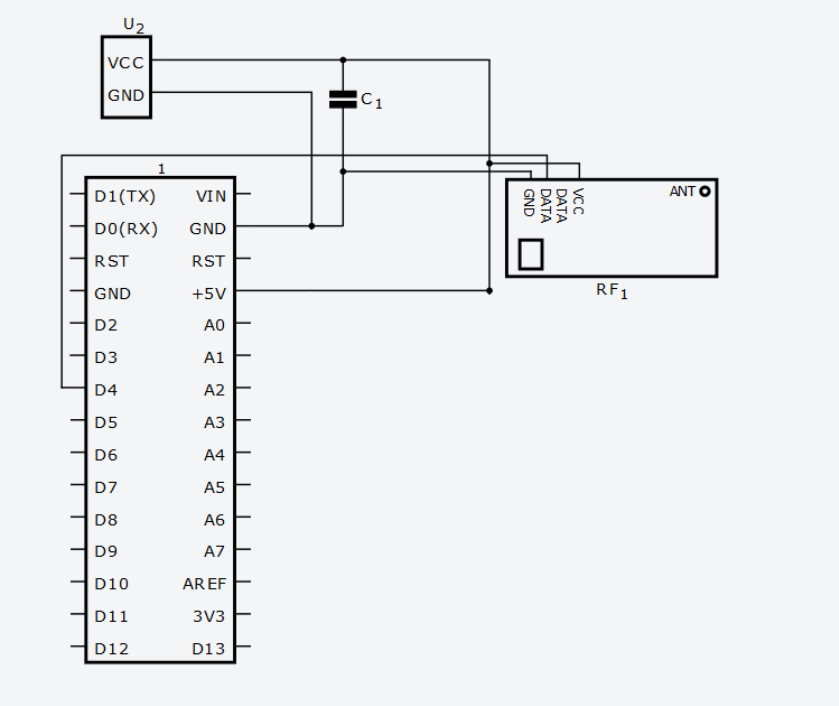

Figura 17 - Ligações do Módulo Arca Frigorifica

Ainda é possível ver também um condensador(C1) para maior estabilidade da alimentação do circuito.

Após isto foi introduzido o sensor de temperatura DS18B20 como já referido anteriormente (Figura 18). O sensor precisa de uma resistência(R1) de 4.7K ohm na sua linha de dados.

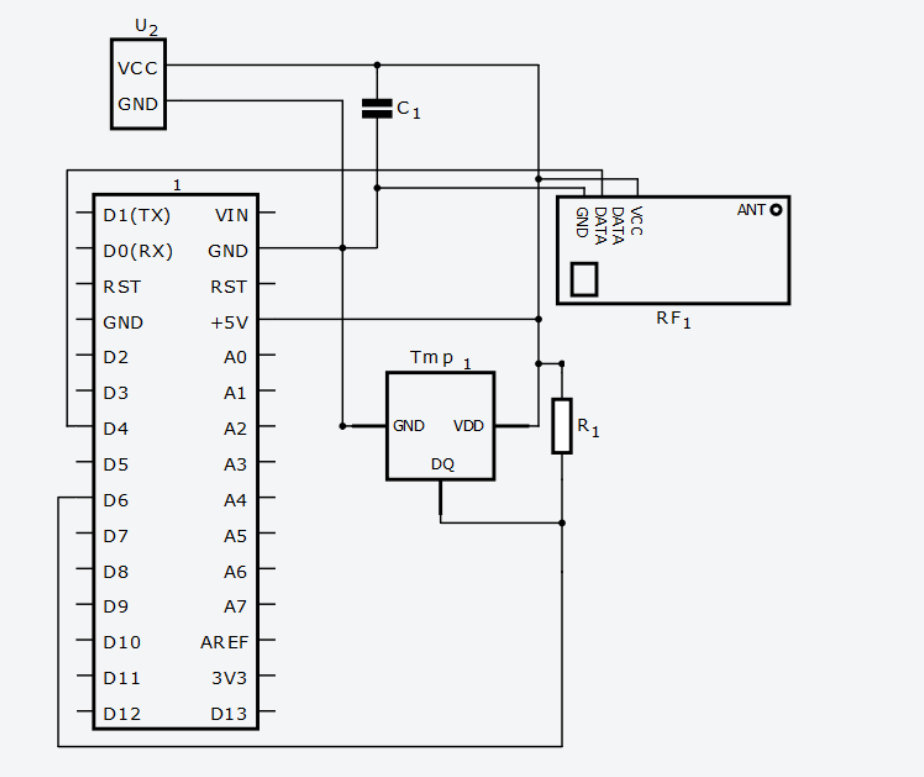

Figura 18 - Ligações do Módulo Arca Frigorifica

Por fim, é necessário adicionar o controlo para a arca frigorífica, esse controlo será feito por um Relé (U1), onde a sua bobine trabalha a 5V e os seus contactos suportam 240VAC (Figura 19).

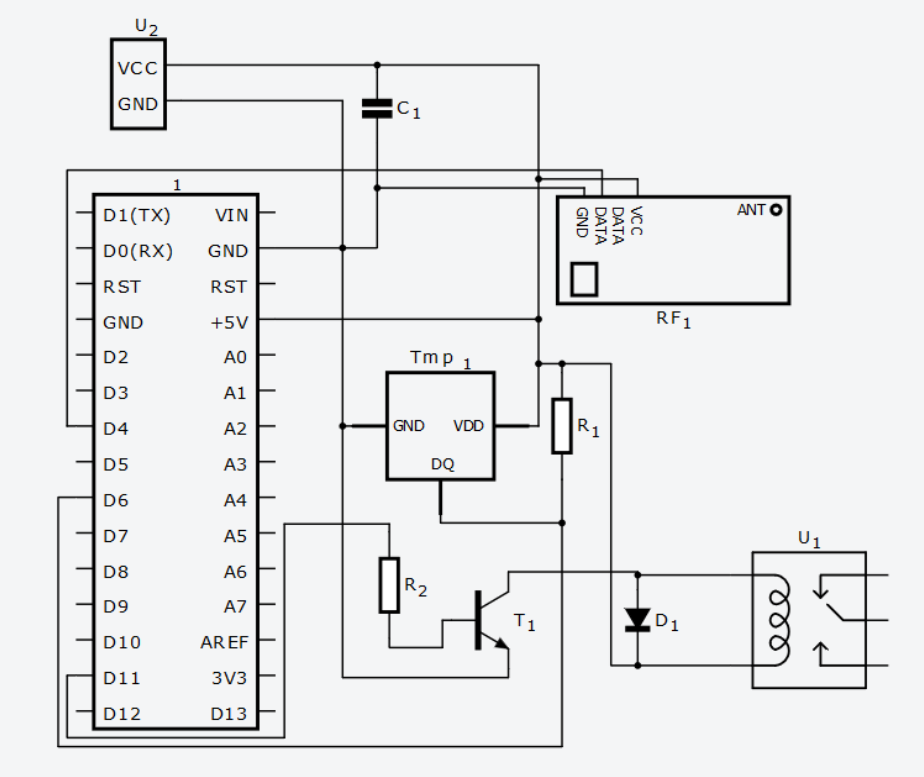

Figura 19 - Ligações do Módulo Arca Frigorifica

Visto que o Arduíno não consegue fornecer corrente suficiente para ativar o relé (U1) é necessário utilizar um transístor, neste caso foi utilizado um transístor(T1) do tipo NPN, e por consequência é necessária uma outra resistência(R2) para interligar o Arduíno ao transístor.

Como o relé possui internamente uma bobine que é capaz de gerar uma indução de corrente, é necessário introduzir um díodo (D1) em paralelo com o relé para absorver os picos de tensão criados ao desligar e assim não queimar o transístor.

Para efeitos de teste e demonstração o sensor de temperatura foi substituído por um potenciómetro (R3), ficando assim o esquema conforme a Figura 20. A troca implicou mudanças a nível de hardware pois o sensor comunicava de forma digital e o potenciómetro de forma analógica, e no código também é alterada a função responsável pela leitura de dados.

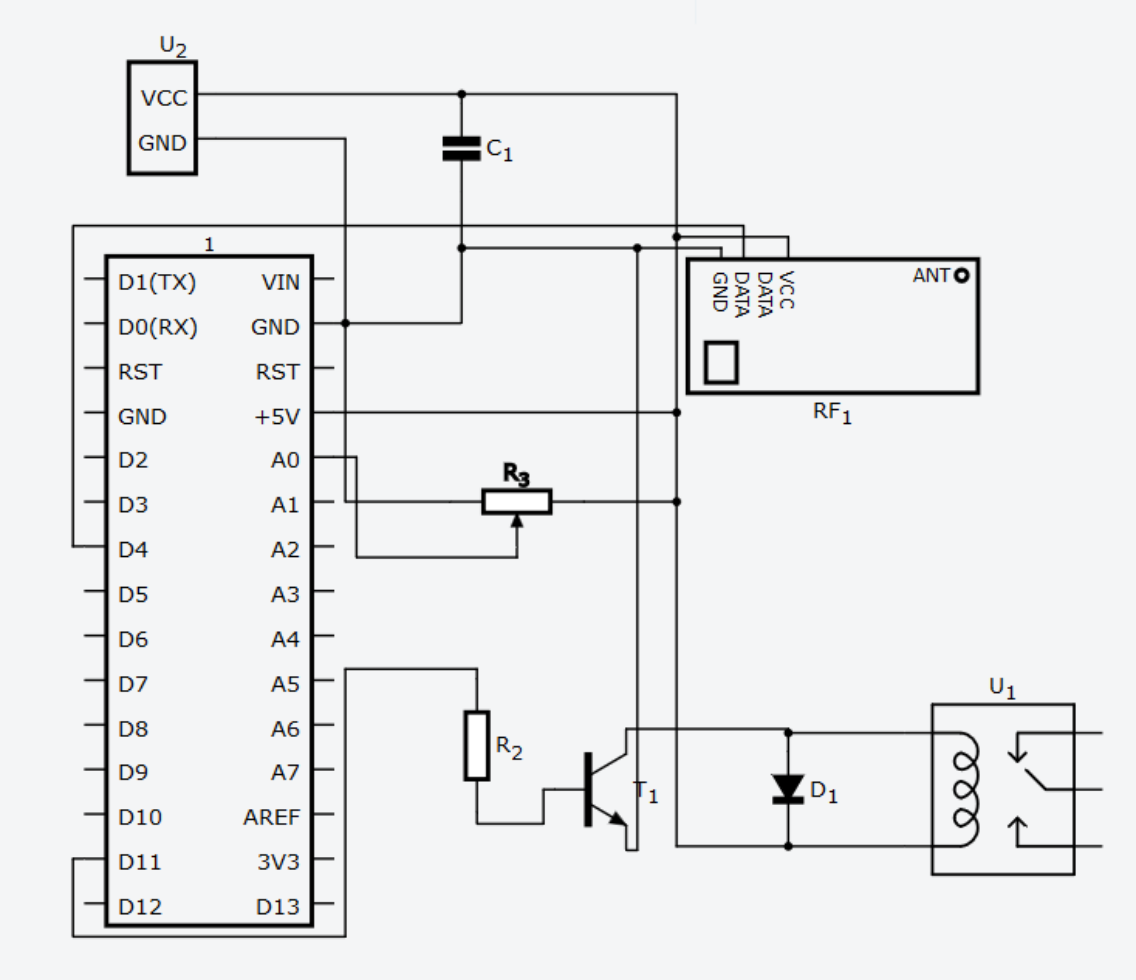

Figura 20 - Módulo Arca Frigorifica completo

O Algoritmo para o módulo é o seguinte:

Função *setup()*

1. Iniciar a porta serie com o PC
2. Configurar o pino do led verde como OUTPUT
3. Configurar o pino de receção de dados
4. Configurar o modo de receção
5. Configurar a velocidade de receção
6. Iniciar a receção de dados

Função *loop()*

1. Se passou o tempo definido
   1.1. Chamar a função *atualiza()*
2. Se existir dados recebidos

2.1.Se o tamanho dos dados for 2 e o primeiro carácter for “1” e o segundo for o “#”
2.1.1. Atualizar para estado Renovável
2.2.Se o tamanho dos dados for 2 e o primeiro carácter for “0” e o segundo for o “#”
2.2.1. Atualizar para estado Normal

Função atualiza()

1. Ler a temperatura
2. Se estiver em Renovável
   2.1. Se a temperatura atual for menor ou igual que a temperatura em renovável menos 1
      2.1.1. Desligar
   2.2. Se a temperatura atual for maior ou igual que a temperatura em renovável mais 1
      2.2.1. Ligar
3. Senão
   3.1.Se a temperatura atual for menor ou igual que a temperatura em normal menos 1
      3.1.1. Desligar
   3.2. Se a temperatura atual for maior ou igual que a temperatura em normal mais 1
      3.2.1. Ligar
4. Incrementar a variável de tempo

Função mandaLigar()

1. Se estiver desligado
   1.1 Se o intervalo de estado for maior que o intervalo mínimo
      1.1.1. Atualizar o pino do led com o valor HIGH
      1.1.2. Atualizar o estado para ligado

Função mandaDesligar()

1. Se estiver ligado
   1.2. Se o intervalo de estado for maior que o intervalo mínimo
      1.2.1. Atualizar o pino do led com o valor LOW
      1.2.2. Atualizar o estado para desligado

Função lerTemperatura()

1. Ler valor do pino A0
2. Converter num valor entre -30 e -5

Na parte do código foi indicada a biblioteca usada para receber dados através de RF

```
#include <VirtualWire.h>
```

Após isso foi definido em que pino está conectado o recetor.

```
#define receive_pin 4
```

Foi também definido o pino do Led do Arduíno a fim de mostrar o estado e também ativar o Relé.

```
#define LedVerde 13
```

Após isso é necessário na função Setup fazer as configurações restantes.

```
void setup() {
  Serial.begin(115200);
  pinMode(LedVerde, OUTPUT);
  vw_set_rx_pin(receive_pin);
  vw_set_ptt_inverted(true);
  vw_setup(2000);
  vw_rx_start();

}
```

Na imagem é possível observar toda a função Setup() onde foi logo iniciada a porta Serial para possível procura de falhas, iniciar o pino de saída do relé/led e todas as configurações para a receção de dados. Para a receção de dados a biblioteca usada precisa de ordem de início (wr_rx_start();) pois utiliza um temporizador interno do microcontrolador para descodificação da mensagem recebida.

Como falado anteriormente, a arca frigorífica no seu funcionamento utiliza partes mecânicas que requerem alguma precaução ao ligar/desligar, para isso foram definidas constantes de tempos de espera assim como para os limites de temperaturas e algumas variáveis.

```
const int tempNormal = -14; //temperatura de funcionamento normal
const int tempActiva = -20; //temperatura de funcionamento em renovavel
const long interval = 1000; //1 segundo de intervalo entre atualizacoes
const int intervaloMinimo = 300; //segundos

bool renovavel = LOW; //variavel de tipo de funcionamento
bool estado = LOW; //variavel de estado funcionamento
int intervaloEstado = 0; //tempo desde a ultima alteracao de estado
int tempAtual = 0;
unsigned long previousMillis = 0;
```

Para uma fácil manipulação do código, foi tudo separado em funções específicas.

No caso de ser necessário ligar ou desligar o funcionamento da arca, foram criadas duas funções.

```
void mandaLigar() {
  if (!estado) {
    if (intervaloEstado > intervaloMinimo) {
      digitalWrite(LedVerde, HIGH);
      estado = HIGH;
      intervaloEstado = 0;
    }
  }
}

void mandaDesligar() {
  if (estado) {
    if (intervaloEstado > intervaloMinimo) {
      digitalWrite(LedVerde, LOW);
      estado = LOW;
      intervaloEstado = 0;
    }
  }
}
```

Nestas funções é possível ver que, antes de realizar qualquer ação, é verificado o estado atual de funcionamento e ainda se já decorreu o intervalo mínimo naquele estado. Após isso é que envia o comando de alterar o estado do pino digital e atualiza as variáveis.

A função responsável por analisar os dados e chamar a função de ligar/desligar é a função atualiza().

```
void atualiza() {
  lerTemperatura();
  if (renovavel) {
    if (tempAtual <= (tempActiva - 1)) {
      mandaDesligar();
    }
    if (tempAtual >= (tempActiva + 1)) {
      mandaLigar();
    }
  }
  else {
    if (tempAtual <= (tempNormal - 1)) {
      mandaDesligar();
    }
    if (tempAtual >= (tempNormal + 1)) {
      mandaLigar();
    }
  }
  intervaloEstado++;
```

Esta função vai fazer a leitura da temperatura, verificar se está em modo de renovável ou não, e após isso verificar se está dentro de algum parâmetro para enviar ordem de ligar ou desligar. Esta função é necessário ser executada uma vez em cada segundo que passe, logo esse trabalho fica a cargo da função loop() onde o Arduíno estará sempre a executar.

```
void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    atualiza();
  }
  uint8_t buf[VW_MAX_MESSAGE_LEN];
  uint8_t buflen = VW_MAX_MESSAGE_LEN;
  if (vw_get_message(buf, &buflen)) {
    int i;
    Serial.print("Recebido: ");
    for (i = 0; i < buflen; i++) {
      Serial.print(buf[i]);
    }
    if (buflen == 2 && buf[0] == 49 && buf[1] == 35)renovavel = HIGH;
    if (buflen == 2 && buf[0] == 48 && buf[1] == 35)renovavel = LOW;

  }
}
```

A função loop() utiliza a função millis() para fazer uma comparação e verificar se já passou o intervalo predefinido de 1 segundo, caso seja verdade, executa a função que atualiza() e retorna.

Seguindo a função loop(), são definidos um array de bytes e uma variável de tamanho, para receber e apresentar a mensagem recebida pela comunicação RF, após a mensagem recebida ela é enviada por porta serial para debug e é atualizada a variável de estado.

De seguida só fica a faltar definir a função de leitura de temperatura. A leitura de temperatura será efetuada por um sensor DS18B20, o que requer incluir bibliotecas.

```
#include <Wire.h>
#include <OneWire.h>
```

O sensor trabalha pelo protocolo OneWire, mas a necessita de funções da biblioteca Wire, por isso é necessário incluir as duas. Após isso, é necessário iniciar a biblioteca com o pino onde está conectado o sensor.

```
OneWire ds(6);
```

Quando é chamada a função de ler a temperatura ela faz todo o trabalho de dar ordem ao sensor para internamente fazer a leitura de temperatura, espera que ele o faça, e posteriormente lê os dados do registo interno do sensor e converte em graus centígrados.

```
void lerTemperatura() {
  byte data[12];
  ds.reset();
  ds.write(0xCC);
  ds.write(0x44);
  delay(760);
  ds.reset();
  ds.write(0xCC);
  ds.write(0xBE);
  for (int i = 0; i < 9; i++) {
    data[i] = ds.read();
  }
  ds.reset();
  tempAtual = ((float)(*((int*)data))/16.0);
}
```

Desta forma sempre que é necessária uma leitura é só executar a função e ela trata de tudo.

Como dito inicialmente, para efeitos de teste e demonstração, este sensor foi substituído por um potenciómetro e o código utilizado foi o seguinte.

```
void lerTemperatura() {
  int sensorValue = analogRead(A0);
  tempAtual = map(sensorValue, 0, 1023, 0, 25) - 30;
}
```

Este código vai ler o valor do potenciómetro, e converter em valores entre -5 e -30 o que é o ideal para testes e demonstração.

Com isto está completo o código do segundo módulo.

6. DESENVOLVIMENTOS COMPLEMANTARES

Este projeto foi desenvolvido com a tecnologia de comunicação RF433MHz devido à facilidade com que se programa o envio de mensagens entre emissor/recetor, mas carece de muitos problemas, desde não ser uma tecnologia com comunicação bidirecional e por isso não existir forma de saber se a mensagem foi entregue, a fraca segurança, etc. No início desde projeto foi abordado o pequeno módulo de comunicação Wi-Fi ESP8266 (Figura 21) que para além das suas funcionalidades Wi-Fi, é possível utilizar o seu microcontrolador interno para execução de algum código. A versão do módulo usada foi a versão ESP-01 que disponibiliza dois pinos nativos de livre programação, sendo eles o GPIO0 e GPIO2.

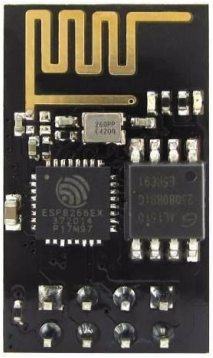

Figura 21 - ESP8266 versão ESP-01

O módulo pode ser programado em ambiente Arduíno, mas possui uma grande desvantagem neste campo, a sua programação requer algum cuidado pois é necessário um conversor usb-serial (o Arduíno possui um na própria placa), e ainda é necessário conectar a placa em modo de programação manualmente. Visto que seria uma boa evolução para o projeto, aprofundou-se um pouco os conhecimentos nesta placa.

Como a placa tem dois pinos IO de livre programação foi pensado duplicação do módulo da arca frigorífica, mas com mais funcionalidades a fim de explorar algumas potencialidades.

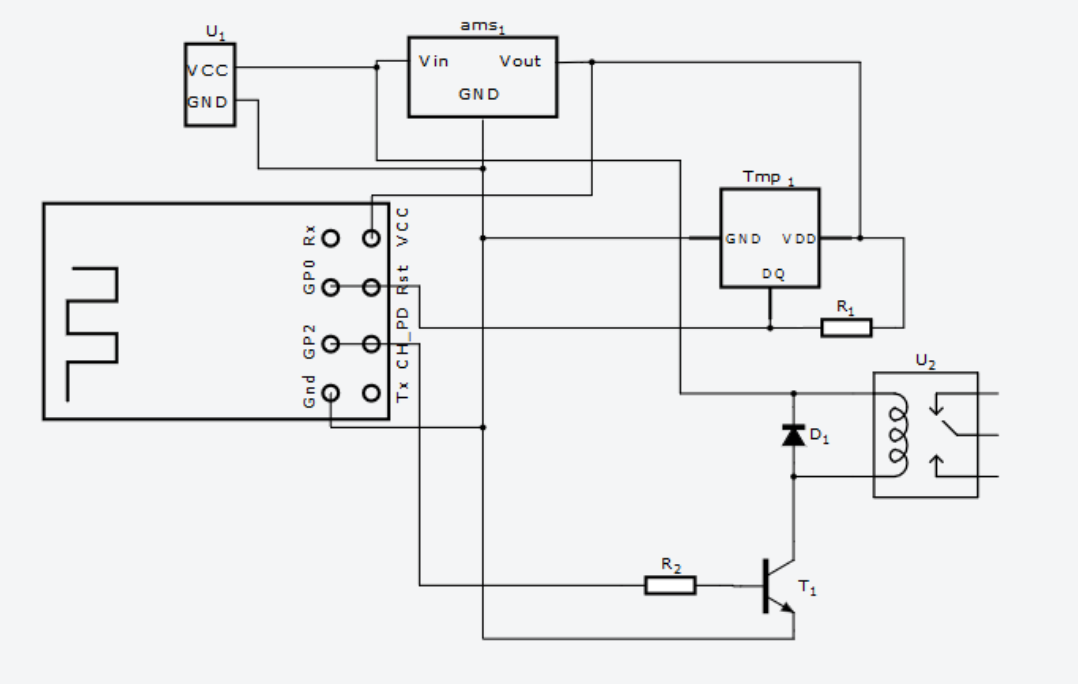

Figura 22 - Esquema de ligações ao ESP8266

Na Figura 22 é possível observar o módulo ESP8266, conectado a ele está o sensor de temperatura DS18B20 (Tmp1) com a sua resistência (R1) de *pull up*, conectado ao pino GPIO0 e também o circuito do Relé (U2), o díodo (D1) de proteção, o transístor(T1) e a sua resistência (R2) conectado ao pino GPIO2. Como o ESP8266 tem a tensão de trabalho de 3.3V é necessário um conversor de voltagem e como antes, foi utilizado o AMS1117 (ams1). O Relé continua a receber os 5V da fonte de alimentação, pois é o único componente que não funciona na gama doas 3.3V.

Visto querer explorar outras funcionalidades, este módulo foi adaptado não a uma arca frigorífica, mas a um simples aquecedor, e terá como funcionalidades, informar a temperatura ambiente, ligar/desligar o aquecedor, e ainda definir a temperatura pretendida.

Utilizando os serviços gratuitos do site ThingSpeak, foi criado um canal de dados (Figura 23) para o efeito, com apenas um campo de entrada de dados, que vai ser preenchido com a temperatura ambiente.

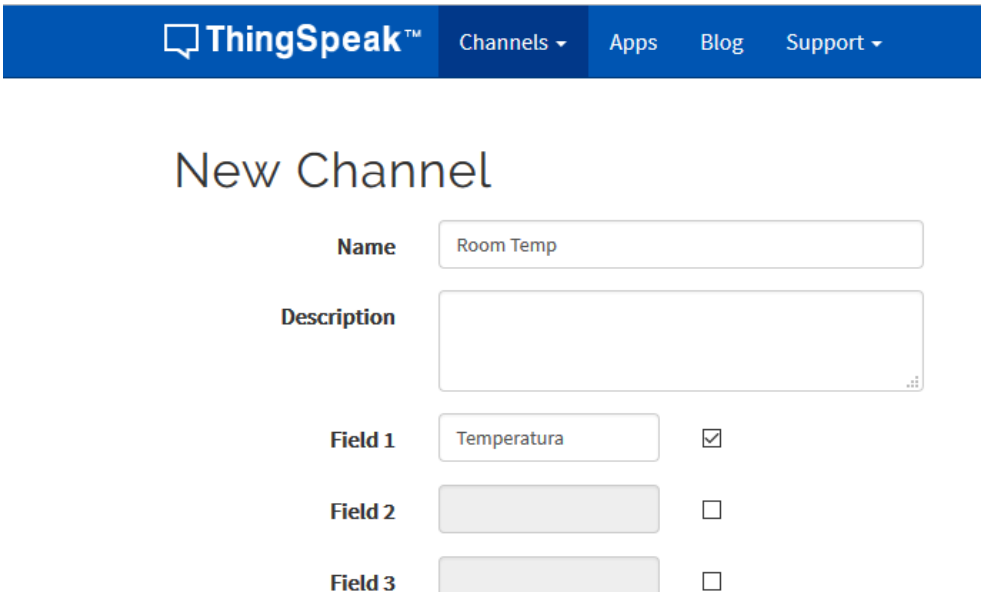

Figura 23 - Criação de canal do ThingSpeak

Ainda outra potencialidade do ThingSpeak é associar ao canal um TalkBack (Figura 24) e assim poder dar ordens ao módulo.

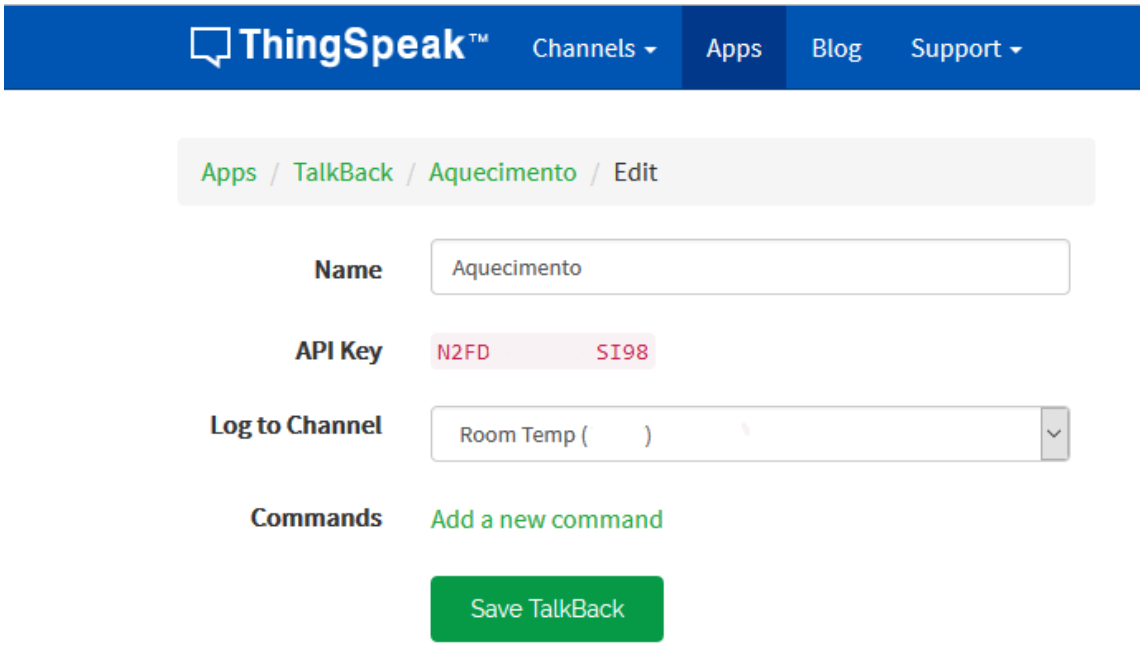

Figura 24 - Criação de TalkBack no ThingSpeak

Continuando no ThingSpeak, e para ficar algo de fácil utilização, foi criado um plugin em HTML e javascript (Java How to Program, 7th Edition) para adicionar ao canal, onde cria as

ordens que o módulo vai receber através do TalkBack, ficando o conjunto no geral como mostra a Figura 25.

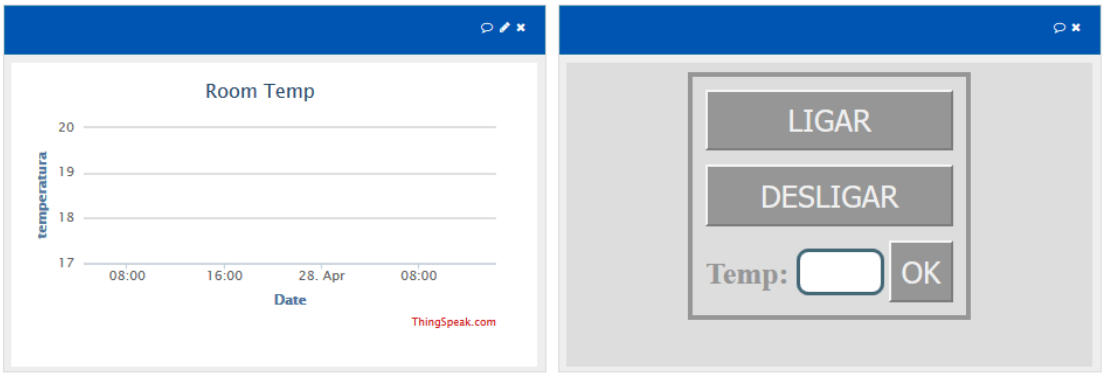

Figura 25 - Aspeto do canal no ThingSpeak

Tendo toda a implementação do lado do servidor concluída, falta criar o algoritmo antes de passar ao código que vai correr no ESP8266.

O algoritmo é o seguinte:

Função setup()

1. Configurar o pino do rele como OUTPUT
2. Desligar o relé
3. Configurar o modo wireless como cliente
4. Conectar a rede wireless
5. Enquanto não ligar a rede
   5.1. Aguardar 500 milissegundos

Funçao loop()

1. Se passou o tempo definido
   1.1.Ler dados
   1.2.Enviar dados
   1.3. Se existir TalkBack
      1.3.1. Se o TalkBack for “TURN_ONN”
         1.3.1.1.Atualiza o estado para ligado
      1.3.2. Se o conteúdo do TalkBack for “TURN_OFF”
         1.3.2.1.Atualiza o estado para desligado
      1.3.3. Se o conteúdo for inteiro maior que 1 e menor que 30
         1.3.3.1.Atualiza a target com o valor inteiro
      1.3.4. Remove o conteúdo do TalkBack

1.4. Se o rele estiver desligado e o estado for ligado e ainda a temperatura atual for inferior 1 grau a temperatura definida
    1.4.1. Ligar o relé
1.5. Se o rele estiver ligado e a variável estado for desligado ou a temperatura atual for maior ou igual a temperatura definida
    1.5.1. Desligar o relé

Função updateThingSpeak()

1. Ler a temperatura
2. Se conseguir conectar ao servidor do ThingSpeak
   2.1. Criar e enviar a String de dados para o Thingspeak
   2.2. Esperar 300 milissegundos
   2.3. Enquanto o cliente tiver dados recebidos para leitura
      2.3.1. Ler um caracter
      2.3.2. Se o caracter for "$"
         2.3.2.1.Ler ate encontrar o fim da linha e guardar como TalkBack
3. Fechar as conexões
4. Sair e devolver a temperatura lida

Função getTemp()

1. Fazer reset na linha de dados
2. Enviar a ordem de ignorar o endereço
3. Enviar a ordem de conversão para o sensor
4. Esperar que o sensor converta a temperatura
5. Fazer reset na linha de dados
6. Enviar a ordem de ignorar o endereço
7. Enviar a ordem para o sensor enviar os dados
8. Ler os dados para o array de dados
9. Fazer reset na linha de dados
10. Devolver a temperatura lida em graus ºC

Posto isto fica a faltar a criação do código. Para ser possível programar para o ESP8266 em ambiente Arduíno é necessário instalar uma extensão ao Arduíno IDE passando ele assim a conseguir compilar código para o ESP8266.

Para começar o código foram incluídas as bibliotecas necessárias.

```
#include <ESP8266WiFi.h>
#include <Wire.h>
#include <OneWire.h>
```

A biblioteca do ESP8266 é obrigatória a sua inclusão, pois é nela que esta presente todas as funcionalidades Wi-Fi da placa, já as bibliotecas seguintes são necessárias para o correto funcionamento do sensor de temperatura. Após isso têm de ser definidas as constantes e variáveis necessárias ao funcionamento do programa.

```
WiFiClient client; //definir o cliente wireless
OneWire ds(0);  //definir o sensor de temperatura
#define Pin 2 // definir o GPIO2 para o rele.

String apiKey = "CVOW[redacted]M7YE"; //API do canal
String talkback_key = "N2FD[redacted]SI98"; //API do Talkback
const char* ssid = "Vo[redacted]2"; //Nome da rede Wireless
const char* password = "[redacted]";  //Password da rede Wireless
const char* server = "api.thingspeak.com";  //endereco do servidor
const long interval = 120000; //tempo entre atualizacoes 120s-2min

unsigned long previousMillis = 0; //variavel para a funçao millis
String TalkBack = ""; //variavel para guardar as respostas do servidor
int target = 18; //valor do limite de temperatura
bool estado = LOW; //estado do aquecedor
```

O código utilizado para o sensor de temperatura foi o mesmo usado no módulo da arca frigorífica, ficando assim só necessário invocar a função quando seja necessário.

```
float getTemp() {
  byte data[12];
  ds.reset();
  ds.write(0xCC);
  ds.write(0x44);
  delay(760);
  ds.reset();
  ds.write(0xCC);
  ds.write(0xBE);
  for (int i = 0; i < 9; i++) {
    data[i] = ds.read();
  }
  ds.reset();
  return ((float)(*((int*)data))/16.0);
}
```

Para uma continuidade de trabalho por funções, foi criada a função necessária para comunicar com o ThingSpeak. Nesta função está tudo o necessário para enviar a temperatura para o servidor, e caso exista algum comando para o módulo ele é recebido e guardado para posterior análise. Para começar é chamada a função do sensor de temperatura para assim obter uma temperatura atualizada.

```
float updateThingSpeak() {
  float t = getTemp();
  if (isnan(t)) {
    Serial.println("falhou a leitura de temperatura!");
    return -1;
  }
```

Após a temperatura obtida é necessário comunicar com o servidor do ThingSpeak, neste passo é necessária alguma atenção. Foram necessárias algumas tentativas para chegar a solução.

```
  if (client.connect(server, 80)) {
    String post = "POST https://api.thingspeak.com/update?api_key=" + apiKey;
    post += "&field1=" + String(t) + "&talkback_key=" + talkback_key + " HTTP/1.0\r\n\r\n";
    client.print(post);
    delay(300);
    String talk;
    while (client.available()) {
      char a = client.read();
      if (a == '$') {
        TalkBack = client.readStringUntil('\r');
      }
    }
    Serial.print("Temperatura: ");
    Serial.print(t);
    Serial.print(" C enviada e talkback: ");
    Serial.println(TalkBack);

  } else {
    Serial.println("Erro ao conectar ThingSpeak");
  }
  client.stop();
  return t;
}
```

Vai ser tentada uma ligação ao servidor para a porta 80, é criada toda a string de dados onde contém a API do canal, a temperatura lida pelo sensor e a API do TalkBack. Após isso faz um pequeno tempo de espera pela resposta do servidor, um pequeno truque usado para facilitar a identificação do texto to TalkBack é inserir em todos os comandos o símbolo “$” no início.

Quando chega a resposta do servidor, procura-se pelo símbolo “$” e fica fácil de identificar qualquer comando. A função retorna o valor da temperatura e termina a sua execução, também com alguns dados enviados pela porta serie para facilitar a perceção de algum erro que surja. Como é normal no ambiente de programação Arduíno, fica a faltar as duas funções principais, sendo elas a função setup() e a função loop().

Na função setup() é necessário inicializar a porta serie, definir o pino do relé como pino de saída, e ainda conectar o módulo Wi-Fi ao router.

```
void setup() {
  Serial.begin(115200);
  delay(1000);
  pinMode(Pin, OUTPUT);
  digitalWrite(Pin, LOW);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Ligado a rede");
}
```

Para conectar a rede Wi-Fi é necessário definir que queremos utilizar o ESP8266 em modo de Cliente wireless, pois ele tem outros modos de funcionamento, após isso é dizer a que rede e password ele se deve ligar e aguardar a sua ligação. Estando esta função completa fica a faltar a função loop(). Nessa função é onde o módulo estará sempre a correr caso não tenha mais nenhuma tarefa para executar.

Como o ThingSpeak tem um intervalo mínimo de espera entre o envio de dados de 60 segundos, e também como um simples aquecedor não necessita de ordens muito rápidas, a atualização de estado foi definida para ser executada cada 2 minutos (120 segundos), assim sendo, foi utilizada a função millis() para garantir que só executa o código uma vez a casa 120 segundos.

```
void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    float tempLida = updateThingSpeak();
```

Apos o ThingSpeak atualizado é a altura de verificar a existência de alguma resposta para o módulo.

```
if (TalkBack != NULL) {
  if (TalkBack == "TURN_ONN") {
    estado = HIGH;
    TalkBack = "";
  }
  if (TalkBack == "TURN_OFF") {
    estado = LOW;
    TalkBack = "";
  }
  int aux = TalkBack.toInt();
  if ( aux >= 1 && aux <= 30 ) {
    target = aux;
    TalkBack = "";
  }
}
```

Caso exista resposta é identificada é realizada a ação necessária.

Como o módulo funciona como um termostato, mesmo que seja enviada a ordem de ligar ele não vai ligar o aquecedor caso esteja uma temperatura ambiente igual ou superior a definida. O aquecedor só liga caso a temperatura ambiente seja inferior a 1ºC relativamente á temperatura definida. Para isso são necessárias duas comparações, a comparação para a temperatura e a comparação para a existência de ordem de ligar.

```
      if (!digitalRead(Pin) && estado && ((target - 1) >= tempLida)) {
        digitalWrite(Pin, HIGH);
      }
      if (digitalRead(Pin) && (!estado || target <= tempLida)) {
        digitalWrite(Pin, LOW);
      }
    }
  }
```

Posto isto está completo o código relativamente ao módulo.

7. CONCLUSÃO

O tema deste projeto foi escolhido por mim devido à minha curiosidade em temas diversos, conhecia este problema e tinha uma ideia própria de como o tentar resolver. Aproveitei alguns conhecimentos adquiridos ao longo do curso para implementar uma solução para o problema. Ao longo do projeto foram encontradas bastantes dificuldades, que me serviram para alargar os meus conhecimentos a vários níveis.

Após a conclusão do projeto pode-se verificar que a ideia inicial foi totalmente implementada, os objetivos foram cumpridos. O projeto traz ainda vantagens aos atuais sistemas fotovoltaicos face às soluções já existentes.

É uma solução relativamente fácil de implementar pois não são necessárias grandes alterações aos sistemas existentes, utiliza um eletrodoméstico existente na grande maioria das habitações e a maior vantagem é de ser um sistema modular que comunica com tecnologia sem fios o que permite a implementação dos diferentes módulos em sítios distintos.

A ideia inicial seria apenas a criação do módulo principal e do módulo da arca frigorifica, mas para verificar a viabilidade do projeto foi criado mais um módulo para a recolha de dados reais para posterior análise que também cumpriu os objetivos.

Com os dados obtidos foi possível ver que existia uma margem generosa na arca frigorífica para se poder aplicar o conceito do projeto pois a arca frigorifica só trabalha perto de 9:27 horas ao longo de um dia, o que é bom pois a exposição solar de Portugal que varia entre 5 a 11 horas.

Visto que ficou de sobra o ESP8266 após a obtenção de dados, achei por bem também implementar algo útil com ele, aumentando o meu nível de conhecimento a nível da “Internet das Coisas”, podendo num futuro passar a ser utilizado no projeto inicial em vez do Arduíno, tendo como principal vantagem a comunicação com redes Wi-Fi (com internet ou não) e assim aumentar o leque de funcionalidades.

Este pequeno projeto foi vantajoso pois tive de relembrar todos os conhecimentos adquiridos de HTML e Javascript para elaborar o Plugin no servidor do ThingSpeak e também conhecer uma nova plataforma. No final compensou todo o trabalho, ficou a funcionar como previsto, sendo possível remotamente saber a temperatura ambiente atual, ligar ou desligar e

ainda programar a temperatura desejada. É fascinante como atualmente é possível criar um modulo eletrónico de pequenas dimensões totalmente funcional.

Como esta solução ainda não existe no mercado, com a ajuda do IPG foi iniciado um pedido de patente, estando a aguardar o resultado.

8. BIBLIOGRAFIA

alldatasheet.com. (s.d.). Obtido de http://www.alldatasheet.com/

Arduino cc. (s.d.). Obtido de https://www.arduino.cc

Java How to Program, 7th Edition. (s.d.).

Nova Energia. (s.d.). Obtido de http://www.novaenergia.net

Solar Shop. (s.d.). Obtido de http://www.solarshop.pt/

VirtualWire Library. (s.d.). Obtido de https://www.pjrc.com/teensy/td_libs_VirtualWire.html

WeatherOnline. (s.d.). Obtido de http://www.weatheronline.pt/weather/maps